High-Throughput Routing for Multi-Hop Wireless Networks

by

Douglas S. J. De Couto

S.B., Computer Science and Engineering (1998)
M.Eng., Electrical Engineering and Computer Science (1998)
Massachusetts Institute of Technology

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

June 2004

Signature of author _____

Department of Electrical Engineering and Computer Science
30 April 2004

Certified by _____

Robert T. Morris
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by _____

Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

High-Throughput Routing for Multi-Hop Wireless Networks

by

Douglas S. J. De Couto

Submitted to the Department of Electrical Engineering and Computer Science on 30 April 2004 in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

The *expected transmission count* (ETX) metric is a new route metric for finding high-throughput paths in multi-hop wireless networks. The ETX of a path is the expected total number of packet transmissions (including retransmissions) required to successfully deliver a packet along that path. For practical networks, paths with the minimum ETX have the highest throughput. The ETX metric incorporates the effects of link loss ratios, asymmetry in the loss ratios between the two directions of each link, and interference among the successive links of a path. Busy networks that use the ETX route metric will also maximize total network throughput.

We describe the design and implementation of ETX as a metric for the DSDV and DSR routing protocols, as well as modifications to DSDV and DSR which make them work well with ETX. Measurements taken from a 29-node 802.11b test-bed show that using ETX improves performance significantly over the widely-used minimum hop-count metric. For long paths the throughput increase is often a factor of two or more, suggesting that ETX will become more useful as networks grow larger and paths become longer.

We also present a simple model for predicting how packet delivery ratio varies with packet size, and detailed measurements which characterize the test-bed's distribution of link delivery ratios and route throughputs.

Thesis Supervisor: Robert T. Morris
Title: Associate Professor of Electrical Engineering and Computer Science

For Ann and Roderick

hark! a wireless
machine reboots silently
beep floppy zz-zzt.
—*M.B.T.*

Contents

1 Introduction **15**
- 1.1 Multi-hop Wireless Networks . . . 15
  - 1.1.1 Antennas . . . 17
  - 1.1.2 Why Not Cellular? . . . 19
  - 1.1.3 The Problem . . . 21
- 1.2 Design Constraints . . . 22
- 1.3 Contributions of this Work . . . 22
- 1.4 How to Read This Dissertation . . . 23

2 Overview of 802.11 Radios **25**
- 2.1 Physical Layer . . . 26
- 2.2 MAC Layer . . . 27
  - 2.2.1 Medium Access . . . 27
  - 2.2.2 Retransmissions and Packet Timing . . . 27

3 The Throughput Problem **31**
- 3.1 Experimental Test-Bed . . . 32
- 3.2 Path Throughputs . . . 33
- 3.3 Distribution of Path Throughputs . . . 35
- 3.4 Distribution of Link Loss Ratios . . . 38

4 Wireless Model **41**
- 4.1 Digital Packet Radios . . . 41
- 4.2 Channel Model . . . 42
  - 4.2.1 Path Loss . . . 43
  - 4.2.2 Multipath . . . 43
  - 4.2.3 Noise . . . 43
  - 4.2.4 Asymmetry . . . 44

4.3 Effect of Spread-Spectrum 45
4.4 Error Model 47
4.4.1 Model Inaccuracies 49
4.4.2 Model Evaluation 50

5 Design of ETX **55**
5.1 ETX Intuition 55
5.2 Design Criteria 56
5.3 The ETX Metric 56
5.3.1 ETX Assumptions 59
5.4 Alternative Metric Designs 60

6 Protocol Implementation **63**
6.1 Operation of DSDV 63
6.2 Changes to DSDV 65
6.3 DSR Implementation 66
6.4 Router Configuration Details 69
6.5 Modular Route Metrics 70

7 ETX Evaluation **73**
7.1 Routing Protocol Tests 73
7.1.1 Experimental Setup 74
7.1.2 DSDV Performance 75
7.1.3 DSR Performance 85
7.1.4 ETX versus ‘Best’ 87
7.2 Static Throughput Tests 90
7.3 Single Link Tests 92
7.4 Evaluation Summary 95

8 Future Directions **97**
8.1 ETX Improvements 97
8.2 Wireless Routing and ETX 99

9 Related Work **103**

10 Conclusion **109**

List of Figures

1-1 A multi-hop ‘mesh’ wireless network. 16
1-2 Directional vs. omnidirectional antennas. 18
1-3 A cellular wireless network. 20

2-1 Barker spreading sequence used by 802.11. 26
2-2 802.11 packet formats. 28
2-3 802.11 packet timing diagrams. 28

3-1 The test-bed. 32
3-2 DSDV with minimum hop-count finds low-throughput routes. . . 34
3-3 Hop count does not predict throughput. 36
3-4 Measured throughput of all static routes. 37
3-5 One-hop packet delivery ratios between each pair of nodes. 38

4-1 The hidden terminal problem. 45
4-2 A Rake receiver. 46
4-3 Predicted loss ratios for a few links. 51
4-4 Predicted versus actual delivery ratio for all links. 53
4-5 Distribution of delivery ratio prediction errors by packet size. . . . 54

5-1 Signal strength does not predict delivery ratios. 62

6-1 DSDV pseudo-code. 67
6-2 DSDV queuing configuration. 70
6-3 Generic metric abstraction. 71
6-4 Link measurement interface. 72

7-1 ETX finds higher throughput routes than minimum hop-count. . . 76
7-2 Per-pair ETX throughput vs. hop-count throughput. 77
7-3 ETX vs. minimum hop-count TCP throughput. 79

7-4 ETX throughput for large packets. 81
7-5 ETX throughput at 30 mW transmit power. 82
7-6 ETX finds higher throughput routes than link handshaking. 83
7-7 Throughput effect of DSDV delay-use modification. 84
7-8 Throughput of DSR and ETX, without transmission feedback. . . 85
7-9 Throughout of DSR with ETX, with transmission feedback. 86
7-10 ETX mispredictions. 88
7-11 Real transmission counts predict route throughput. 89
7-12 Route throughputs can change quickly. 91
7-13 ETX vs. transmission count over single links. 93
7-14 ETX vs. transmission count over single links, big packets. 94
7-15 Link delivery ratios can change quickly. 96

Acknowledgments

Some graduates write terse acknowledgments thanking exactly the select few who sped them along their way. Others ramble on, trying to give thanks to everyone who helped throughout the years of toil, only to glance at their dissertation two weeks later and realize they left out key individuals.

You've probably already guessed this is more like the second than the first. This work took me a long time, and I had a lot of help.

I am most grateful to my advisor Robert Morris, who shepherded this work through the lengthy stages of implementation, debugging, experimentation and writing. He held me to the highest standards of quality and accuracy, and any value in my work is thanks to those high standards. I am also grateful for his frank opinions and advice, and for listening when I was frustrated.

Daniel Aguayo made significant contributions to this work from the beginning. He helped deploy the first version of the test-bed network, designed and ran many of the earlier experiments which motivate this work, and wrote the DSR routing protocol implementation that I used. I am indebted to Dan for countless technical discussions, and for keeping my feet on the ground.

John Bicket's non-stop hacking made possible the second version of the test-bed network, which connected every node to the wired Ethernet. His work made debugging and running experiments at least an order of magnitude easier. John also ran many experiments.

Eddie Kohler created the Click modular router, which allowed elegant implementations of ETX and the routing protocols. Click was especially useful for debugging, testing, and simulating my protocol implementations. Click is an extremely powerful tool, and I recommend it to anyone who feels the need to implement network protocols. Eddie graciously spent many hours fixing Click so that it could do everything I needed. Eddie also wrote the xwrits typing break program, without which my wrists and back would hurt very much.

Jinyang Li's work on the capacity of multi-hop routes was an important result that led to the idea of the ETX metric. She's also fun to work with!

I want to thank everyone in the Parallel and Distributed Operating Systems research group and on the former fifth floor at LCS for making it such a stimulating place to work and learn. Chuck Blake, Frank Dabek, and Dave Andersen were especially helpful, and I've learned a lot from them. My office mates deserve special thanks for putting up with me over the years, and for being patient with my questions: Benjie Chen, Emil Sit, and Emmett Witchel.

I also owe a very special thanks to all the lab members who graciously agreed to host a test-bed node in their office. They put up with me tramping in and out of their offices for over two years, crawling under their desks, and making the machines beep and grind their disks when I rebooted the network several times a day. The poem in the front of this dissertation is dedicated to them.

Life, however, is not all about the lab, and I was lucky enough to have many good friends to remind me of that. Thank you Michael Taylor for the movies and a ready ear, Howard Davis for the good times and perspective on finishing the Ph.D., and Cove Johnstone for listening and giving good advice. Danielle Ames, Kenny Benet, Claudio Cairoli, Sue Foight, Suzanne Garb, Jon Ted Lendon, and Victor Preciado: thank you for the sunny days, fresh breezes, cold beverages, and hot tubs in Marblehead. You kept me sane.

Finally, my family supported me over the years, even though I didn't always know how to explain what I was trying to do.

My graduate program was partially supported by the Sir John W. Cox scholarship from the Bank of Bermuda.

Chapter 1

Introduction

This work describes how to find high-throughput routes in multi-hop wireless packet networks. Using the *expected transmission count* (ETX) metric presented here, routing protocols can find multi-hop routes that have up to twice the throughput of those found using the minimum hop-count metric. Most routing protocols minimize the hop-count[1] metric, which is the number of wireless links in a route, regardless of the performance of each link. Since multi-hop wireless networks likely contain many lossy links, routes preferred by the hop-count metric also often contain lossy links, which reduce throughput. The ETX metric is based on the loss ratio of each link in a route, as well as the number of links in a route. Because it prefers shorter routes with better links, ETX selects high-throughput routes.

Throughput is not the only property that network users care about. For example, voice and interactive users prefer low delay, while video users want to minimize jitter, which is the variability in delay and throughput. However, these applications and many others benefit from increased throughput, which is the focus of this work.

1.1 Multi-hop Wireless Networks

A multi-hop wireless network is a network of computers and devices (*nodes*) which are connected by wireless communication *links*. The links are most often implemented with digital packet radios. Because each radio link has a limited communications range, many pairs of nodes cannot communicate directly, and must forward data to each other via one or more cooperating intermediate nodes.

[1] We will often use 'hop-count metric' to mean the minimum hop-count metric.

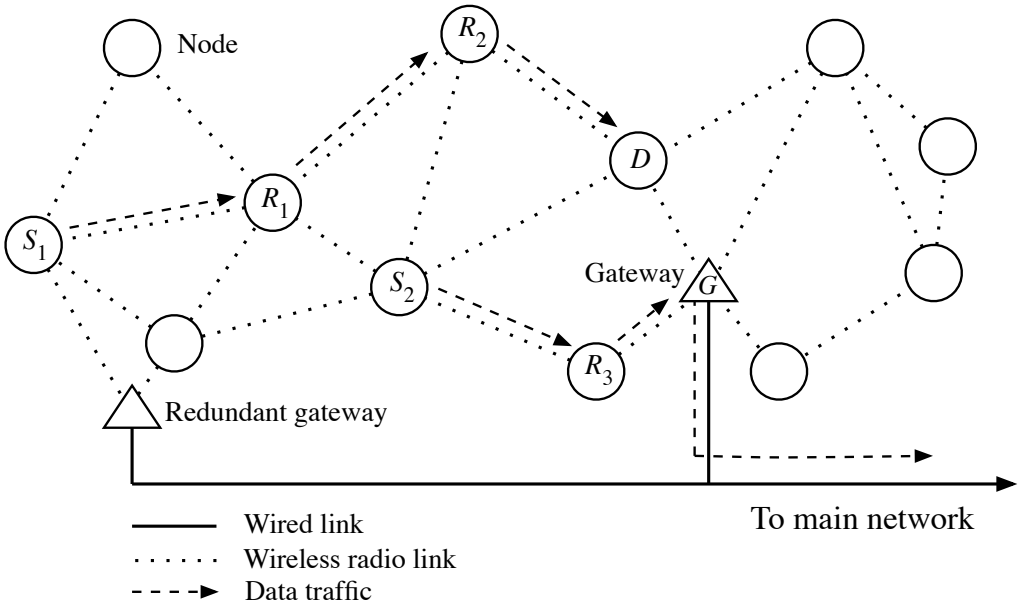


Figure 1-1: A multi-hop ‘mesh’ wireless network. Node S_1 sends data to node D via cooperating nodes R_1 and R_2, while node S_2 sends data out of the network via node R_3 and the gateway G.

A source node transmits a packet to a neighboring node with which it can communicate directly. The neighboring node in turn transmits the packet to one of its neighbors, and so on until the packet is transmitted to its ultimate destination. Each link that a packet is sent over is referred to as a *hop*; the set of links that a packet travels over from the source to the destination is called a *route* or *path*. Routes are discovered by running a distributed *routing protocol* on the network. Figure 1-1 shows an example of a multi-hop wireless network. These networks are often called 'mesh' networks, in reference to the topology formed by the links and nodes. Typically a mesh network does not operate in isolation, and often has one or more gateways that connect it to a larger internet.

1.1.1 Antennas

Wireless networks can be built using *omnidirectional* antennas, *directional* antennas, or some combination of the two. An omnidirectional antenna transmits and receives radio signals equally in all directions, forming links with other nodes in all directions. A directional antenna transmits and receives radio signals in a single direction, only forming links with nodes in that direction. Figure 1-2 illustrates the difference.

Links built using directional antennas can be approximated as wired links, and traditional wired network routing techniques will work well over these links [76]. Because each directional antenna is one end of a single point-to-point link, network designers can individually engineer each point-to-point link to have a very low loss ratio [69]. Also, each link can be considered independently by the routing protocol, because the narrow coverage area of the directional antennas greatly reduces interference between links.[2]

The disadvantage of point-to-point directional links is that they are difficult to install and engineer. Antennas must be aimed, link budgets must be calculated, and the network topology must be determined beforehand, as each link requires its own antenna at each end. To add a new node to the network, the network designer must explicitly decide where to add new links, and explicitly design in redundancy and fault tolerance by adding multiple point-to-point links for each new node.

On the other hand, a node with a single omnidirectional antenna can form multiple links with many other nodes in any direction. The network designer can

[2] Unfortunately, link independence is limited by the non-ideal coverage patterns of real antennas. At close enough ranges, two directional antennas can interfere with each other irregardless of direction.

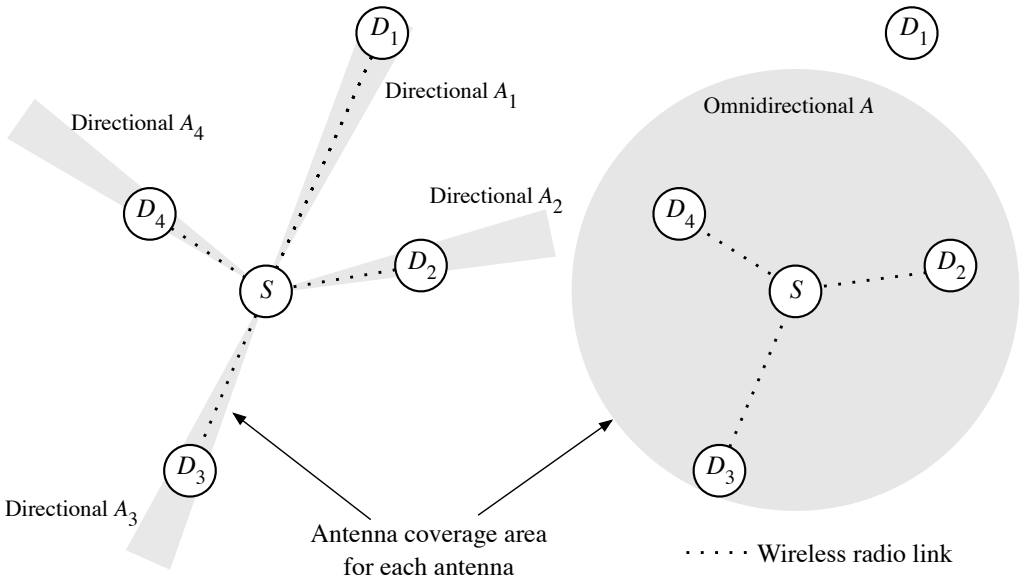


Figure 1-2: Building wireless networks with directional versus omnidirectional antennas. The left side shows links using directional antennas, while the right side shows links using an omnidirectional antenna. In both cases, the network operator would like to build links between node S and each of its neighbors D_1 through D_4. On the left, each directional antenna has a very narrow, long-range coverage area, and Node S has a good link to its neighbors D_2 through D_4. S has a marginal link to D_1, since it as the edge of the antenna range. Each link requires a separate antenna, A_1 through A_4. On the right, node S uses a single omnidirectional antenna A, with a very broad but relatively short-range coverage area. Node S has good links to nodes D_2 and D_4, a marginal link to D_3, and no link to D_1, which is out of range.

easily add a new node by placing it within range of any other node. Because the antenna is omnidirectional, it does not need to be aimed, and forms multiple links simultaneously with any nearby neighbors, providing redundant links with little extra effort.

Although omnidirectional antennas make it easy to deploy new nodes, they have their own drawbacks. Because each antenna is the end-point of multiple links, it is not feasible to independently engineer most of the links in the network, and many links will be lossy. Furthermore, the overlapping antenna coverage patterns of nearby nodes will cause them to interfere with each other, reducing the throughput of each link.

The rest of this work is about how to find high-throughput routes in multi-hop wireless networks built with omnidirectional antennas. Antennas are a significant part of the cost of a multi-hop wireless network, and unlike digital radios, their cost and functionality do not scale according to Moore's law. However, digital radios follow the steeply increasing performance curve of computer processing power, and will continue to become cheaper, with increasingly sophisticated signal processing, coding, and routing capabilities. As a result, systems built with a single omnidirectional antenna at each node will likely remain much cheaper than those built with multiple directional antennas at each node, even as their performance gap narrows.

1.1.2 Why Not Cellular?

A multi-hop wireless network can be expanded by incrementally adding nodes to the network, typically at the edges as its physical area grows. In this sense it is self-expanding: since the network nodes using the network cooperate to provide connectivity to each other, the network exists wherever there are nodes. This is in contrast to a cellular network, where data travels directly from wireless nodes to fixed base stations. Data typically travels from a base station to its destination over a wired network, as shown in Figure 1-3. Since each base station provides a fixed amount of network coverage to a fixed geographical area (the 'cell'), there is only network connectivity where base stations have been predeployed. Cellular base station locations and radio configurations are carefully chosen not to interfere with adjacent cells, while avoiding coverage gaps between cells. Although cellular networks can be incrementally deployed and expanded, the overhead and planning required to setup a base station is much larger that that required to deploy a few extra new nodes in a multi-hop wireless network. In addition, unlike multi-hop wireless networks, installing a cellular base station requires some preexisting or

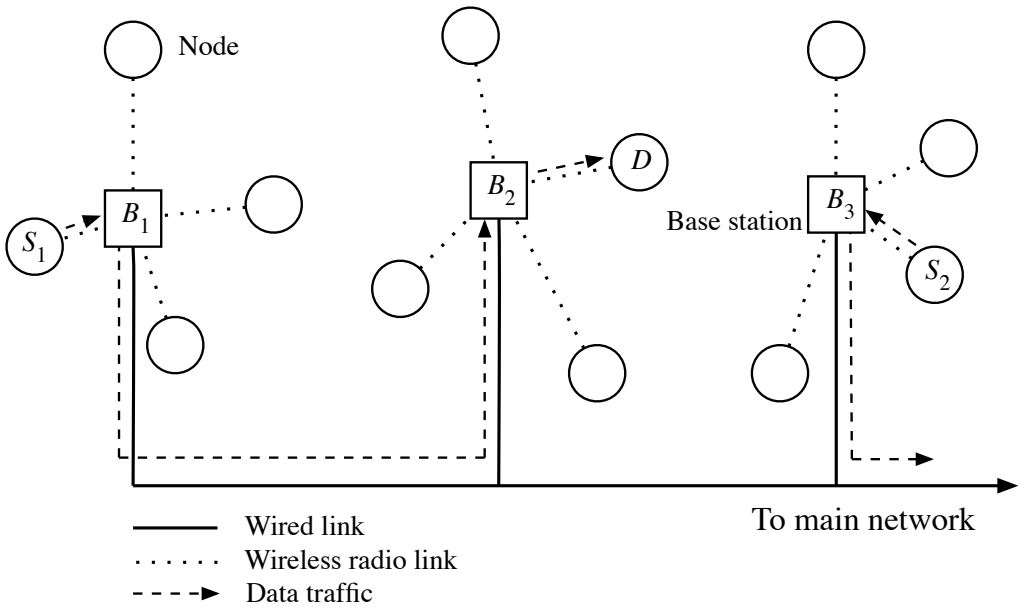


Figure 1-3: A cellular wireless network. Node S_1 sends data to node D via base stations B_1 and B_2, which communicate over the wired network. Nodes S_1 and D might also share the same base station. Node S_2 sends data out of the network via base station B_3.

additional network infrastructure, such as land lines or long distance radio links to obtain network connectivity for the base station.

1.1.3 The Problem

The same flexibility that makes it easy to deploy multi-hop wireless networks with omnidirectional antennas also makes it difficult to find good links and routes. Unlike wired networks or wireless networks with point-to-point wireless links, it is difficult to engineer the communications links. When a new node is deployed, it will form communications links with *all* nodes that are within range, including those that are on the edge of communications range. As discussed in Chapter 4, links at the edge of communications range will have very poor signal strength, and packets sent over these links will often be lost completely, or will be corrupted and discarded at the receiver. Since lost or corrupted packets do not transfer any useful data over the link, the effective bandwidth of a lossy link is less than that of a good link. The percentage of transmitted packets that are lost or discarded is termed the *loss ratio*; its complement, *delivery ratio*, is the percentage of transmitted packets that are successfully received. Some of the links formed by adding a new node to the network will have low delivery ratios and low throughput, some will have high delivery ratios and high throughput, and many will have intermediate delivery ratios and throughput. As Chapter 3 will show, the network as a whole has a broad distribution of link delivery ratios: some good links, some bad links, and many intermediate.

In general, there will be many potential routes between each pair of nodes in the network; because each route uses a different set of links, these routes will have different throughputs. The routing protocol select the route with the highest throughput. Routing protocols use a *route metric* to decide which route to use between a pair of nodes. A route metric is a number assigned to each route; the routing protocol then selects the route with the best metric.[3] The route metric is based on some underlying property of the route. For example, the commonly used hop-count metric is the number of links in a route. Protocols choose a route with the minimum hop-count; there may be many minimum hop-count routes, in which case protocols often choose arbitrarily between them. Chapter 3 shows that an arbitrary minimum hop-count route often has much lower throughput than other routes between the same pair of nodes.

[3] The best metric is typically the smallest metric, but depends on the interpretation of the metric.

1.2 Design Constraints

Multi-hop wireless networks have a very rich design space, and designers must make choices in many dimensions when building these networks. We make several assumptions about the underlying network that constrain the design space.

As discussed above, we assume that the network uses omnidirectional antennas, as they are cheaper and more convenient.

We implicitly assume that the network is a store-and-forward network which decodes and retransmits packets at each hop, according to predetermined routes that are decided by a routing protocol. This is one traditional way of operating data networks, and fits in well with current practice. However, it precludes techniques like network coding [5, 43, 48], which make more efficient use of the underlying network capacity.

We also assume that all network nodes have a single radio and antenna, operate on the same shared channel, and use the same fixed bit-rate and transmission power. But in reality, many radios can reduce link bit-rates for increased reliability, and variable transmission power can be used to trade off transmission range for total network capacity. Some radios can switch between multiple channels, or transmit on multiple frequencies simultaneously, as in orthogonal frequency-division multiplexing (OFDM) [16, 19, 21]. Finally, placing multiple radios and antennas into each node can reduce or eliminate interference between links in the same route. Chapter 8 describes the applicability of ETX when these assumptions are relaxed.

1.3 Contributions of this Work

This main contribution of this work is the design, implementation, and evaluation of the estimated transmission count (ETX) metric, which is designed to enable routing protocols to find high-throughput routes. The ETX of a route is the total number of packet transmissions and retransmissions required to send a packet across the route, assuming that each link in the route retransmits the packet until it is successfully received across the link. ETX is designed for links with link-layer acknowledgments (ACKs) and retransmissions, as provided by IEEE 802.11 radios [18]. The ETX metric for a route is calculated using measurements of the lossiness of each link in the route. Routing protocols select routes with the minimum ETX. For short routes (up to and including 3-hop routes), the minimum-ETX route is the maximum-throughput route; for longer routes, the minimum-

ETX route is still a high-throughput route. The design of the ETX metric does not depend on a particular routing protocol; Chapter 7 shows that ETX improves the throughput of both Dynamic Source Routing (DSR) [37], an on-demand source routing protocol, and Destination-Sequenced Distance-Vector (DSDV) [61] routing, a proactive table-driven distance-vector routing protocol. We also presents a set of design changes and implementation techniques that allow DSR and DSDV to work well with ETX, in Chapter 6.

Additional contributions are a detailed exploration of the performance of minimum hop-count routing on a wireless test-bed using 802.11b radios (Chapter 3), and a simple model of how link loss ratios vary with packet size (Chapter 4). Chapter 3 explains why minimum hop-count often finds routes with significantly less throughput than the best available throughput, and quantifies the throughput difference between the typical minimum hop-count route and the highest throughput route. Chapter 4 shows how to use a few link loss ratio measurements to predict loss ratios at different packet sizes; these predictions can be used to decrease the protocol overhead of the ETX metric, by allowing ETX to measure links with small packets. ETX is also likely improve network capacity.

In order to demonstrate that ETX is effective, Chapter 7 presents measurements taken from the test-bed network. These measurements show that ETX improves the throughput of multi-hop routes by up to a factor of two over the minimum hop-count metric. ETX provides the most improvement for paths with two or more hops, suggesting that ETX offers increased benefit as networks grow larger and paths become longer.

1.4 How to Read This Dissertation

Chapter 2 reviews the 802.11 radios used in this work, and can be skipped by readers familiar with 802.11. Chapter 3 describes the test-bed network and its throughput problems. Chapter 4 explains how digital packet radios work, and gives a simple model of how packet size affects loss ratios; it can also be skipped by readers familiar with digital communications, although they might wish to read Section 4.4 to learn about the packet size model. Chapters 5 and 6 present the design and implementation of ETX and explain how it is used by the routing protocols. Chapter 7 evaluates how well ETX works on a real wireless network, while Chapter 8 shows how ETX might be improved in the future, and how it is useful when the wireless network design space is expanded. Chapter 9 surveys related work in wireless routing. Finally, Chapter 10 concludes.

Chapter 2

Overview of 802.11 Radios

This chapter provides an overview of the IEEE 802.11b radios used in this work. 802.11 is an IEEE standard for the physical and medium access control (MAC) layers of wireless LANs [18]. The standard specifies several layers of a packet radio system, including radio modulation and coding, packet formats, and the MAC protocol for managing contention between multiple senders. The original 802.11 standard specifies radios that can operate at one and two megabits per second; the follow-on 802.11a [19], 802.11b [20] and 802.11g [21] standards specify additional bit-rates and packet formats.

The main focus of the 802.11 standard is networks with a star topology, and almost all 802.11 radios are used this way. In these networks wireless clients exchange packets with specially-designated wireless access points. The access points then relay client packets between the wireless clients and a wired LAN. In this scenario, wireless clients do not exchange packets directly with each other; all packets pass through an access point. This mode of operation is often referred to as *infrastructure* mode.

However, the protocols and experiments described in this work do *not* use the radios in infrastructure mode. Instead, the radios are used in a *peer-to-peer* mode where they can directly send and receive packets from any radio which might be in range. This mode is also sometimes called *ad hoc* mode. The 802.11 standard refers to radios operating in this mode as an Independent Basic Service Set (IBSS).

Bit-Rate	Modulation	Bits/Symbol	Chips/Symbol
1 Mbps	DBPSK	1	11
2 Mbps	QPSK	2	11
5.5 Mbps	CCK	4	8
11 Mbps	CCK	16	8

Table 2.1: IEEE 802.11b bit-rates and their associated modulation.

$$+1, -1, +1, +1, -1, +1, +1, +1, -1, -1, -1$$

Figure 2-1: Barker spreading sequence used by 802.11.

2.1 Physical Layer

The IEEE 802.11 standard describes three physical layers: an infrared layer, a frequency-hopping spread-spectrum layer, and a direct-sequence spread-spectrum (DSSS) layer. Almost all 802.11 radios use the DSSS physical layer, as do the 802.11b radios we used.

The DSSS physical layer specifies how bits and packets are transmitted over the radio air interface. IEEE 802.11b specifies four bit-rates, with associated modulation techniques, as summarized in Table 2.1.

After modulation, the data symbols are encoded by an 11-chip Barker spreading sequence, at 11 megachips per second. The Barker sequence used is shown in Figure 2-1. Table 2.1 shows how many chips encode each symbol for each bit-rate.

In the United States, 802.11 and 802.11b specify 11 channel center frequencies, starting at 2,412 MHz, and spaced 5 MHz apart. Since after spreading with the Barker code, the main lobe of the transmitted signal has a frequency width of 22 MHz, these channels actually overlap significantly with each other. However, it is possible to choose three channels without significant overlap.

Each 802.11 packet transmission consists of a 148-bit preamble and a 48-bit physical layer header, followed by the 802.11 payload. The preamble and physical layer header bits are sent at 1 Mbps, and the 802.11 payload bits can be sent at any of the 802.11 bit-rates. The contents of the preamble are specified by the 802.11 standard. The physical layer header specifies the total length of the packet and the bit-rate used for the 802.11 payload. The 802.11b standard specifies additional optimizations that decrease the time required for the preamble and physical layer header when higher bit-rates are used for the payload; this reduces packet overhead at 802.11b's 5.5 and 11 Mbps data rates.

2.2 MAC Layer

The 802.11 MAC protocol is a carrier-sense multiple-access scheme with collision avoidance (CSMA/CA). The standard refers to this scheme as the distributed coordination function (DCF). The goal of the MAC protocol is to allow multiple competing senders to share the radio medium without interfering with each other.

2.2.1 Medium Access

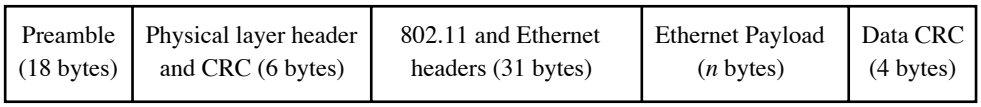
Before sending a packet, a potential sender listens to see if any other transmission is in progress. If there is no such transmission, or once such a transmission is over, the sender waits for a mandatory of time called the DCF interframe space (DIFS). After the DIFS time has passed, the sender chooses a random back-off time *b* from its *contention window*. The contention window has a minimum length of 620 microseconds, and a maximum length of 2,460 microseconds. After each successful transmission, the contention window is set to its minimum value; after each failed transmission, the contention window is doubled, up to the maximum value. The sender waits for the back-off time *b* to pass before attempting to send its packet. If some other radio transmits while the sender is waiting for *b* to elapse, the sender does not count that time as waiting, and resumes waiting at the end of the transmission. That is, the sender waits for *b* amount of *idle* medium time before attempting to send.

The 802.11 standard also specifies an optional request-to-send/clear-to-send (RTS/CTS) protocol which can further reduce radio contention in some scenarios. As RTS/CTS is not used in this work, we do not describe it further.

2.2.2 Retransmissions and Packet Timing

The 802.11 MAC supports two kinds of data packets: broadcast and unicast. Broadcast packet are intended to be received by any radio which hears them them, and are delivered to the networking layer on that radio's node. Unicast packets are directed to a specific destination node. When a radio receives an unicast data packet directed to it, it immediately sends back an acknowledgment (ACK) packet after a short interframe space (SIFS), and delivers the incoming data packet to the networking layer. Other radios may receive the same unicast packet, but they discard it and do not send an ACK response. Each packet includes a destination address so that a radio can decide if the packet was intended for it. Figure 2-2 shows the formats of data and ACK packets.

(a) 802.11 data frame, 59 + *n* bytes over the air.



(b) 802.11 ACK frame, 38 bytes over the air.

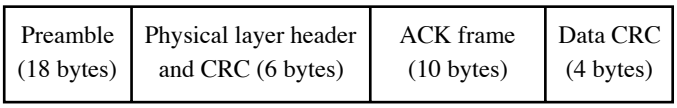
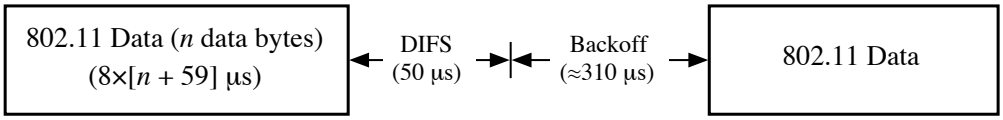


Figure 2-2: Packet formats for 802.11 data and acknowledgment packets.

(a) Packet timing for 802.11 broadcasts.



(b) Packet timing for 802.11 unicasts.

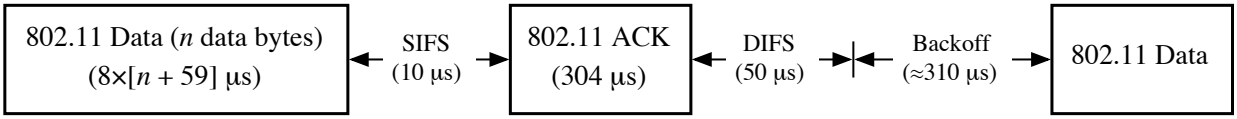


Figure 2-3: Packet timing diagram for 802.11 data traffic, assuming no contention for the radio channel. The total time required to send an 802.11 data broadcast at 1 Mbps with a *n*-byte data payload is $8 \times [n + 59] + 50 + 310 = 832 + 8n$ microseconds. The total time for an unicast is increased by 10 + 304 microseconds because of the ACK packet, and is 1,146 + 8*n* microseconds.

If an unicast sender does not receive an ACK packet after a specified period of time (SIFS + DIFS time after sending the data packet), it marks the transmission as failed. The sender then increases its back-off window, enters back-off, and tries to resend the packet. A sender will repeatedly try to retransmit a packet up to a specified maximum number of tries[1] giving up and discarding the packet.

Figure 2-3 shows the packet exchanges and timings for broadcast and unicast packets at 1 Mbps, assuming that every packet transmission is successful and that there is no contention. The figure shows the average expected back-off time of 310 microseconds. In the absence of contention, the back-off window should be at its minimum size of 620 microseconds, and the average expected random back-off is one-half of that. The maximum broadcast and unicast throughputs of a given packet size can be calculated in packets per second by inverting the time required to send a single packet. For example, for the 134-byte payload used throughout this work, the unicast throughput B can be calculated as

$$B = \frac{1}{1,146 + 8 \times 134} = 451$$

packets per second. For unicast packets with a 1,386-byte payload, the throughput is 82 packets per second.

[1]The radios we used have a default of a maximum of 16 retries.

Chapter 3

The Throughput Problem

Most existing wireless routing protocols use the minimum hop-count route metric: they select routes with the fewest links. The minimum hop-count metric implicitly assumes that links either work well, or do not work at all, and that all working links are equivalent. Furthermore, most protocols assume links that deliver routing control packets such as DSDV route updates or DSR route queries will also successfully deliver data packets.

However, these assumptions are incorrect for multi-hop wireless networks with omnidirectional antennas. Unlike wired and wireless networks with point-to-point links, where the performance of each link can be tightly controlled and engineered, networks with omnidirectional antennas have many wireless links with a wide range of intermediate loss ratios. These lossy links are not useful for data, but deliver enough routing control packets so that the routing protocol uses the link. Measurements in Section 3.4 illustrate the even distribution of link loss ratios for an indoor 802.11 test-bed; others have measured similarly even distributions for an outdoor 802.11 network [4], and for indoor and outdoor sensor networks [12, 78, 80].

Given a broad variation in link loss ratios, hop-count will choose links poorly. This is because minimizing the hop-count of a route maximizes the distance traveled by each hop, which reduces the received signal strength and increases the loss ratio. Even if the best route is a minimum hop-count route, there may be many routes with the same minimum hop-count, but with widely varying qualities. The arbitrary choice made by minimum hop-count is not guaranteed to be the highest-throughput route. This chapter shows that minimum hop-count routing typically finds routes with significantly lower throughput than the best available, using measurements of the DSDV routing protocol on a test-bed network. We ex-

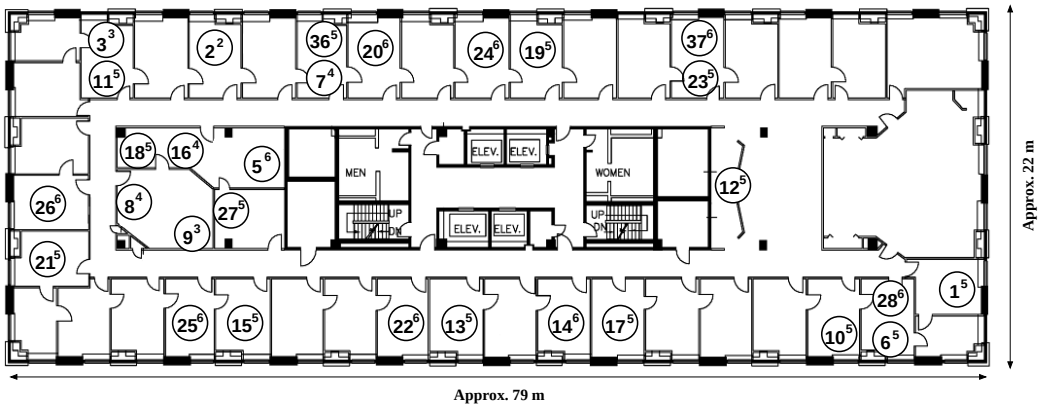


Figure 3-1: A map of the test-bed. Each circle is a node; the large number is the node identifier, and the superscript indicates which floor of the building the node is on.

plain why minimum hop-count does poorly by looking at the distribution of route throughputs and link loss ratios.

3.1 Experimental Test-Bed

All the data in this chapter are the result of measurements taken on a 29-node wireless test-bed. Each node consists of a stationary PC with a Cisco/Aironet 340 PCI 802.11b [18] card and an omnidirectional 2.2 dBi dipole antenna, also called a ‘rubber duck’ antenna. Each PC runs the Linux operating system. The nodes are placed in offices and lounges on five consecutive floors of an office building. Their positions are shown in Figure 3-1.

The test-bed runs new implementations of the DSDV and DSR routing protocols, described in Chapter 6.

The 802.11b cards are set to transmit at one megabit per second (Mbps) with one milliwatt (mW) of transmit power. RTS/CTS is turned off, and the cards are set to ‘ad hoc’ (IBSS, DCF) mode. Each data packet in the following measurements consists of 24 bytes of 802.11b preamble, 31 bytes of 802.11b and Ethernet encapsulation header, 134 bytes of data payload, and 4 bytes of frame check sequence: 193 bytes in total. An 802.11b ACK packet takes 304 microseconds to transmit, the inter-frame gap is 60 microseconds, and the minimum expected mandatory back-off time is 310 microseconds, resulting in a total time of 2,218

microseconds per data packet. This gives a maximum throughput of 451 unicast packets per second over a loss-free link.

While the test-bed itself carried only the data and control traffic involved in each experiment, interference of various kinds was inevitably present. In particular, each floor of the building has four 802.11b access points, on various channels.

The test-bed was designed to experiment with wireless routing protocol implementations, and is one of the larger 802.11-based multi-hop wireless test-beds currently described in the research literature [2, 3, 76, 34, 15, 50, 51]. There are also many commercial multi-hop wireless networks, which are not publicly documented; some of these are smaller than the test-bed described here, but many are much larger, both in the number of nodes and in the area covered by the network. We believe that although radio link performance is known to be quite different indoors than outdoors [69], the conclusions drawn from measurements of this test-bed are still valid for many other networks. This is for two reasons: first, initial measurements of a larger outdoor rooftop network corroborate many of the findings described in this chapter [4]; and second, the main effects we observe stem from the underlying network design, not the specific performance of any particular link.

3.2 Path Throughputs

Figure 3-2 compares the throughput of routes found with a minimum hop-count metric to the throughput of the best static routes that could be found. Each curve shows the throughput cumulative distribution function (CDF) for 100 node pairs; the pairs are randomly selected from the $29 \times 28 = 812$ total ordered pairs in the test-bed. A point's x value indicates the throughput between the pair, in packets per second; the y value indicates the fraction of pairs with less throughput. The left curve is the throughput CDF achieved by routing data using DSDV with the minimum hop-count metric. The right curve is the throughput CDF for the best known path between each pair of nodes. Packets were only sent between one pair at a time. That is, there was no data cross traffic. For each pair, the DSDV and best-path tests were run immediately after one another, to limit variation in link conditions over time.

The 'best' static route between each pair of nodes was found by sending data along ten potential best paths, one at a time, and selecting the path with the highest throughput for each pair. Potential best paths were identified by running an off-line routing algorithm, whose inputs were measurements of per-link loss ratios similar

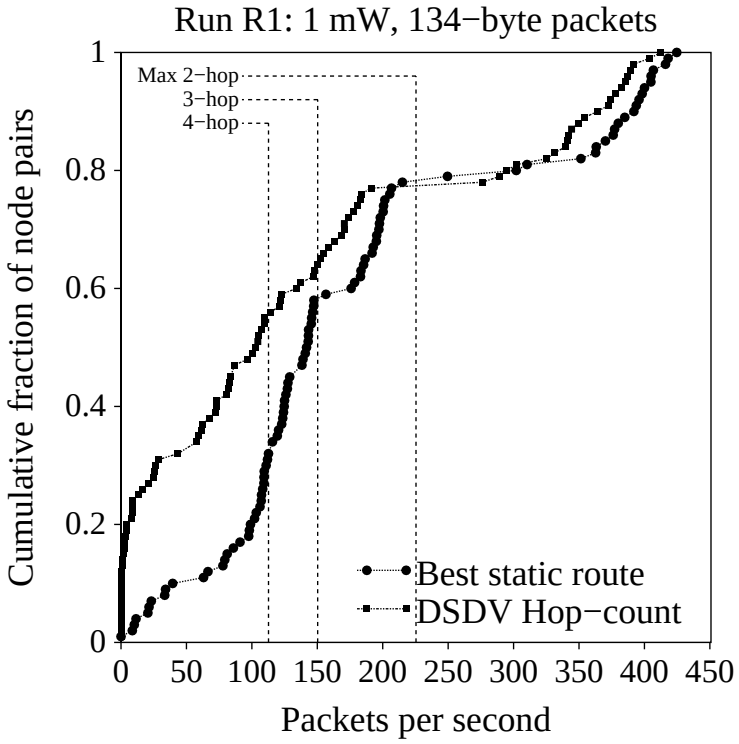


Figure 3-2: When using the minimum hop-count metric, DSDV chooses paths with far less throughput than the best available routes. Each line is a throughput CDF for the same 100 randomly selected node pairs. The left curve is the throughput CDF of DSDV with minimum hop-count. The right curve is the CDF of the best throughput between each pair, found by trying a number of promising paths. The average throughput difference is 42 packets per second (σ = 61). The vertical lines mark the theoretical maximum throughput for routes of each hop-count.

to those in Section 3.4. The algorithm also incorporated a penalty to reflect the reduction in throughput caused by interference between successive hops of multi-hop paths. New link measurements were collected roughly every hour during the experiment; the best paths for each pair were generated using the most recently available loss data. Each node ran a user-level program which forwarded packets according to source routes in the packet headers.

The throughputs in Figure 3-2 are split into two main ranges, above and below 225 packets per second. Pairs with throughputs above 225 sent data along single-hop paths; pairs with throughputs at or below 225 sent data over multi-hop paths or very poor single-hop paths. Multi-hop paths have less throughput because transmissions on the successive hops interfere with each other. In a two-hop path, the middle node cannot receive a packet from the first node at the same time it is sending a packet to the last node, limiting throughput to one-half the link throughput. Similar effects cause the fastest three-hop route to have a capacity of about $450/3 = 150$ packets per second [47].

Minimum hop-count performs well whenever the shortest route is also the fastest route, especially when there is a one-hop link with a low loss ratio. A one-hop link with a loss ratio of less than 50% will outperform any other route. This is the case for all the points in the right half of Figure 3-2. Note that the overhead of DSDV route advertisements reduces the maximum link capacity by about 15 to 25 packets per second, which is clearly visible in this part of the graph.

The left half of the graph shows what happens when minimum hop-count has a choice among a number of multi-hop routes. In these cases, the hop-count metric usually picks a route significantly slower than the best known. The most extreme cases are the points at the far left, in which minimum hop-count is getting a throughput close to zero, and the best known route has a throughput of about 100 packets per second. The minimum hop-count routes are slow because they include links with high loss ratios, which cause bandwidth to be consumed by retransmissions. The zero-throughput points on the left are due to asymmetry: DSDV with hop-count chose asymmetric links which delivered routing packets in the reverse direction, but no data packets in the forward direction.

3.3 Distribution of Path Throughputs

Figure 3.3 illustrates a typical case in which minimum hop-count routing would not favor the highest-throughput route. The figure shows the throughputs of several static routes from node 23 to node 36. The routes are the eight highest-

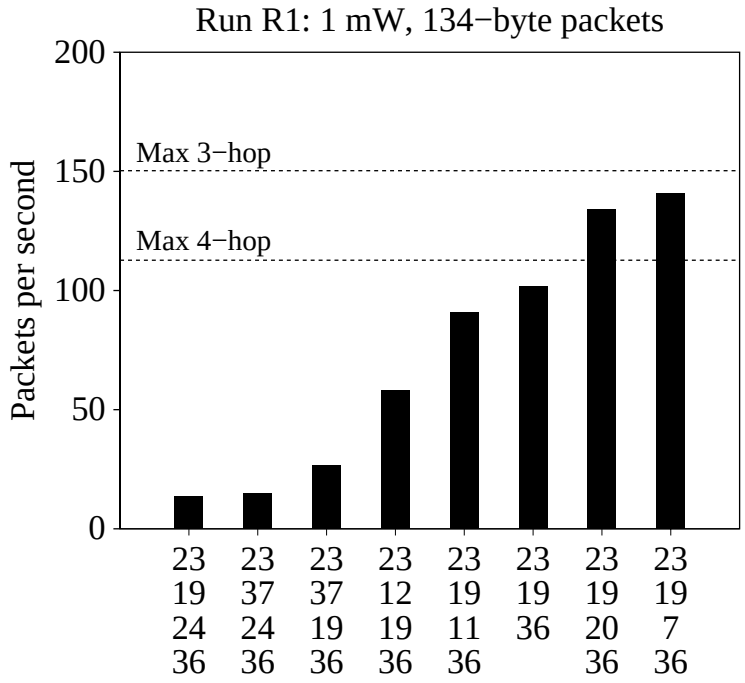


Figure 3-3: Hop count does not predict throughput. This graphs shows the measured throughputs from node 23 to node 36, along the eight highest-throughput routes found in the ‘best’ static route tests. The minimum hop-count route does not have the highest throughput, and there are many three-hop routes with very different throughputs.

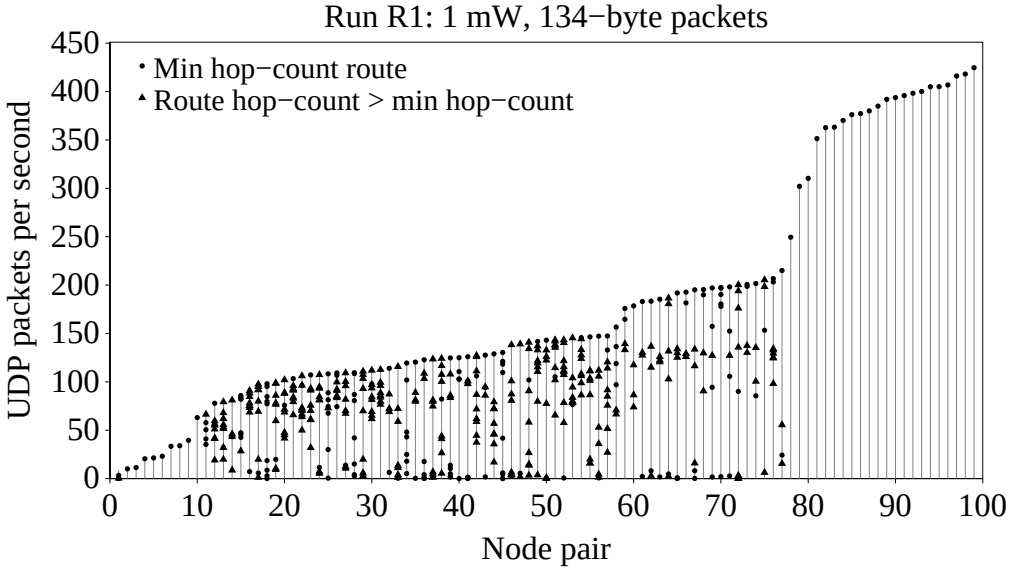


Figure 3-4: Measured throughput of all static routes. Circles mark the throughput of minimum hop-count routes; longer routes have their throughput marked with triangles. 99 pairs are shown here; a minimum hop-count route had the highest throughput on 73 of those pairs. Multi-hop routes were not tested for pairs with a one-hop throughput of greater than 225 packets per second, as that is faster than any multi-hop route can deliver packets.

throughput routes between 23 and 36 which were found in the ‘best’ static route experiments described above. The graph shows that the shortest path, a two-hop route through node 19, does not yield the highest throughput. The best route is three hops long, but there are a number of available three-hop routes which provide widely varying performance.

Figure 3-4 shows the ‘best’ static route results for all the node pairs tested. Although the fastest route many pairs was a minimum hop-count route, 35 pairs have multiple minimum hop-count routes, typically with very different throughputs. Furthermore, the minimum hop-count route was not the fastest route for a quarter of the pairs. A routing protocol that selects randomly from the shortest hop-count routes is unlikely to make the best choice, particularly as the network grows and the number of possible paths between a given pair increases.

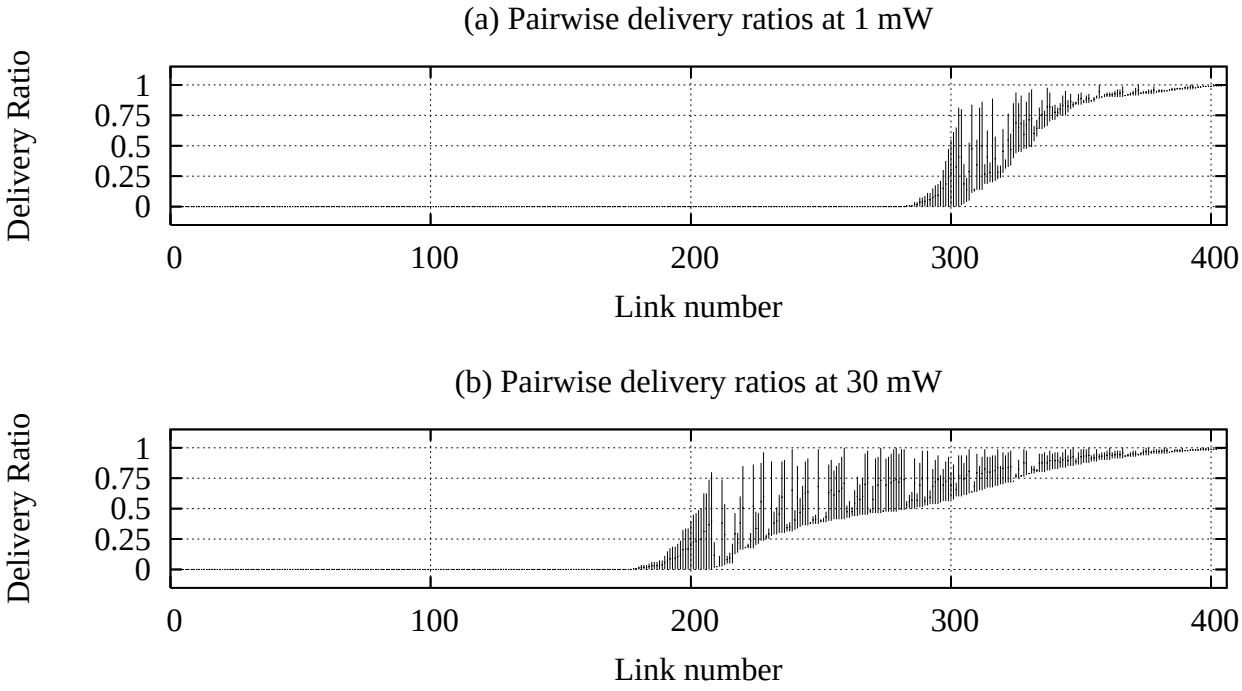


Figure 3-5: One-hop packet delivery ratios between each pair of nodes at 1 mW (above) and 30 mW (below). The top and bottom ends of each vertical line indicate the delivery ratios in the two directions. The bars in each graph are sorted by the minimum of the two directions, so the link numbers do not necessarily match between the two graphs. The packet size is 134 bytes of 802.11b data payload. Data for all 406 pairs of hosts are shown. Many links are asymmetric, and there is a wide range of loss ratios.

3.4 Distribution of Link Loss Ratios

Figure 3-5 shows the underlying delivery ratios of each link in the network, which helps explain why high-throughput paths are difficult to find. Each vertical bar corresponds to the direct radio link between a pair of nodes; the two ends of the bar mark the broadcast packet delivery ratio in the two directions between the nodes. To measure delivery ratios, each node took a turn sending a series of broadcast packets for two seconds, and counted the number of packets that the radio reported as transmitted. Packets contained 134 bytes of 802.11b data payload, and were sent at a rate of 40 packets per second. Every other node recorded the number of packets received. The delivery ratio from node X to each node Y is calculated by

dividing the number of packets received at Y by the number sent by X. The loss ratio of a link is one minus its delivery ratio. We use the term 'ratio' instead of 'rate' to avoid confusion with throughput delivery rates, which are expressed in packets per second.

Note that 802.11 broadcasts don't involve acknowledgments or retransmissions. Because 802.11 retransmits lost unicast packets, a link's unicast packet loss ratio at higher layers is potentially far lower than the underlying broadcast loss ratio, depending on the maximum number of retransmissions allowed. Since only one node was broadcasting at a time in the network, any packet losses are due to interference from the environment or from transmitters outside the network.

Figure 3-5 has three important features. First, a large fraction of the links have an intermediate delivery ratio in at least one direction. That is, they are likely to deliver some routing protocol packets, but would lose many packets if used for data. Second, there is a full spectrum of link delivery ratios, so some advantage can be expected from making fine-grained choices between links when choosing paths. Third, many links have asymmetric delivery ratios.

As discussed in Chapter 1, using omnidirectional antennas makes it easy to deploy a wireless network, but hard to engineer any particular link to have a very low loss ratio. As a result, many of the links in the network are operating in situations they were not designed for, and therefore have non-negligible loss ratios. These links are operating with low SNRs, high noise, and excessive multipath due to the wide variety of obstacles indoor, such as doors, walls, furniture, and people.

The network has a wide range of link loss ratios because the links are operating in a wide range of conditions, despite being in the same network. For example, there is a wide distribution of link distances, and therefore a wide range of received signal levels. This is further compounded by the different levels of receiver noise for each link and the various obstacles blocking and reflecting each link's signal; these produce a wide range of SNR levels throughout the network. Also, the different obstacles to each link produce different multipath effects, further affecting loss ratios in unpredictable ways. Multipath effects are discussed further in Chapter 4.

Even though the effects of attenuation and multipath should be symmetric for each link, many of the links are asymmetric. There are a few explanations for this asymmetry. As just described, receiver noise levels affect the SNR and therefore loss ratios; since each receiver is in a different environment it is likely to have a different noise level and SNR. Some receivers will be in high-noise environments, producing very asymmetric links. Second, although the radios used in the test-bed are identical models, they may come from different manufacturing batches, with slightly different components. Even though the radios were set to use the same

power, their actual power outputs may vary, producing differences in the received signal level at each end of the link. Finally, because the test-bed radios are half-duplex the measurements in each direction of a link occur at different times. It is possible that link conditions changed between the measurements in each direction. For example, a door may have been closed, or a person may have moved their chair; these effects have been informally observed to affect link measurements in the test-bed. Although these time variations may seem to make the link measurements less reliable, they are in fact an an accurate reflection of the sorts of link behavior that a wireless routing protocol will encounter. In a real network, perceived asymmetry will occur as a result of link changes over time. The routing protocol chooses routes before sending data over them, by using protocol packets sent in the reverse direction; links may change between the time protocol packets are sent in the reverse direction and data packets are sent in the forward direction.

Of the 406 node pairs in Figure 3-5a (1 mW), there are 124 with links which delivered packets in at least one direction. Of those links, 28 are asymmetric, with forward and reverse delivery ratios that differ by at least 25%. The 28 asymmetric links involve 22 different nodes, indicating that asymmetry is prevalent throughout the whole network, and not isolated to only a few nodes and links. Because 802.11b uses link-level acknowledgments (ACKs) to confirm delivery, both directions of a link must work well in order to avoid retransmissions. Since most nodes in the network are involved in at least one asymmetric link, routing protocols must cope with asymmetry to be effective.

Figure 3-5b shows similar data, but with the radios set to the 30 mW transmit power, which is about a 15 dB increase in transmit power. As a result, 229 links deliver packets, almost twice as many as in the 1 mW experiment. Also, many more links have very high delivery ratios: at 1 mW there are 69 links (17% of all links) that deliver at least 95% of their packets; at 30 mW there are 121 such links (30% of all links). However, the fraction of working links with high delivery ratios is about the same in both experiments, at just over one half. There are still a large number of asymmetric links at the higher power: 76 links are asymmetric, and 28 nodes are end-points for at least one asymmetric link. This is about 33% of the non-zero links, while only 23% of the non-zero links were asymmetric in the 1 mW experiment. These measurements illustrate that turning up the transmit power does not eliminate the variations in link delivery ratios across the network. Although increased transmit power will increase the delivery ratio of any particular link, it will also add new non-zero links to the network; these new links will be marginal, with intermediate delivery ratios, and the overall shape of the network's delivery ratio distribution will probably stay the same.

Chapter 4

Wireless Model

This chapter gives a simplified description of how digital radios transmit and receive data packets, along with a description of the sorts of problems radios face when transmitting packets. The purpose of this chapter is two-fold: first, to give a rough sense of why the packet losses described in Chapter 3 occur, and second, to explain the experimentally observed fact that packet loss probabilities vary with the size of the packet. As we will see in later chapters, the accuracy of the ETX metric proposed in Chapters 5 and 6 can be improved by properly accounting for packet sizes. This chapter describes a model that accurately predicts loss ratios at different packet sizes based on the measured loss ratios at two other sizes. Since the model is based on the operation of digital packet radios, we start with a description of how radios work.

4.1 Digital Packet Radios

This section provides a brief outline of how a data packet is transmitted as a radio frequency (RF) signal, and how that signal is converted back to bits at the receiver. For a thorough description, see a standard text such as Sklar [73], Proakis [65], or Rappaport [69].

There are essentially three main steps in transmitting the bits in a packet: *coding* and *modulation*, together with packet *framing*. Coding converts the stream of bits in the packet into a stream of *symbols*; modulation converts each symbol into a RF waveform which is then transmitted. Framing is the process of grouping bits into packets and transmitting them with extra information, which is used by the receiver to know when to start demodulation. Demodulation converts the

stream of RF signals into symbols, which are then decoded into bits. Although coding, modulation, and framing are logically separate steps, radio designs often interleave parts of each step.

There are many different types of coding schemes; they are typically designed to make the resulting signals more robust to problems in the RF channel. One relevant effect of many codes is that multiple adjacent bits in a packet may be grouped together into one symbol. That is, one symbol represents multiple bits, such as two or four bits. Because some coding schemes effectively scramble the bits in a packet, bits that are coded into the same symbol may not be near to each other in the original packet.

Just as there are many sorts of coding schemes, there are many modulation schemes. The modulation scheme describes what sort of RF signal is sent for each symbol. Some schemes indicate which symbol is sent by changing the amplitude of the signal, some by changing the frequency or phase of the signal, and some by a combination of all three techniques. No matter what modulation scheme is used, however, the demodulation scheme needs to know where to look for each symbol in the incoming RF signal in order to correctly demodulate it. That is, the receiver must know when in time each symbol starts and ends. Because packet radio systems are typically asynchronous, a radio may receive a packet at any time, and symbol timing information must be re-established for each packet. This is done by adding extra framing information to each packet, such as a *preamble*. A preamble is a predetermined sequence of symbols transmitted at the beginning of each packet. Since the receiver knows what preamble to look for, it can adjust its symbol timing until it finds the expected preamble; at this point the receiver knows it is receiving a packet, as well as where the symbol boundaries lie.

4.2 Channel Model

The RF signal travels from the transmitter to the receiver over the RF *channel*. The channel could be a cable, free space, obstacles, or some combination of the three. The *channel model* describes how the RF signal is affected by the channel. In general, a channel has two main characteristics: path loss and delay. In addition to these two characteristics, the receiver's version of the signal is affected by noise, which is received in addition to the transmitted signal. Although noise is not strictly part of the channel model, we consider it here as it also affects wireless link behavior.

4.2.1 Path Loss

The transmitter's RF output does not reach the receiver in its original form. The amplitude of an RF signal decreases with distance, as the signal spreads out in space. This attenuation is typically on the order of d^{-2} to d^{-4} for a distance d, depending on the environment (e.g. free space or in-building) and the radio frequencies being used [69]. Receivers will receive weaker signals on longer links. In addition, there may be obstacles blocking parts of the transmitter's signal, such as walls or foliage, which will further attenuate the signal seen at the receiver. Finally, loss in radio hardware such as cables and connectors can also decrease the power of the received signal. The total attenuation is referred to as path loss, and is typically constant over time for a given radio link, assuming that neither end is moving, and that the environment is also static.

4.2.2 Multipath

In addition to path loss, a transmitter's signal may be subject to *multipath* effects. When an RF signal is reflected by obstacles, copies of the signal travel to the receiver over multiple paths simultaneously. In general, each of these paths will have a different path loss, and since each path will be a different length, each path will have a different delay. The net result is that the receiver will see several copies of the transmitter's signal, each with a different magnitude and delay. These shifted copies of the transmitted signal will combine together, either reinforcing or degrading each other.

Because the behavior of multipath effects depends greatly on the exact details of the environment, small (or large) changes in the environment or in the locations of the receiver or transmitter can cause the received signal to vary suddenly over time. This variation generally occurs in mobile radio systems, but can also occur in static networks. For example, obstacles such as people, vehicles, doors, or leaves may move in and out of the way of signal paths.

4.2.3 Noise

The receiver will see RF signals from sources besides the link's transmitter. Since these signals are not carrying information from the transmitter, they are referred to as noise. A common assumption is that the noise is additive white Gaussian noise (AWGN). AWGN has three main features. First, because it is *white noise*, its power is uniform across the whole radio spectrum; that is, the noise has the

same amount of energy, on average, in all frequency bands. Second, white noise is uncorrelated in time; the noise during one time period cannot be predicted from the noise during a previous time period. Finally, because the noise is *additive*, it is simply summed with the transmitter's signal (as modified by the channel) at the receiver. Multiple Gaussian noise sources can be added together to form a single Gaussian noise source.

In real systems there are often many sources of noise that is not AWGN. For example, other transmitters may be using the same radio spectrum. Noise from these transmitters would not be white: it would be focused in one part of the spectrum, and correlated in time. Machinery such as cooling fans or microwaves may also produce predictable time-dependent noise. However, in this chapter, we will only consider AWGN.

Finally, transmissions from adjacent radios in the same network can add noise to an RF link. In many multi-hop wireless networks, all the radios use the same coding and modulation, and transmissions from adjacent radios are likely to have an RF signal strength on the same order as the local link. This sort of interference can be particularly damaging because network traffic patterns make the interference highly correlated. To avoid intra-network interference, most wireless networks use a *medium access control* (MAC) protocol to coordinate adjacent transmissions in the same network. MAC protocols, like that used by 802.11 [18], include mechanisms to prevent the *hidden terminal* problem illustrated in Figure 4-1. They often precede each data transmission with a Ready-to-send/Clear-to-send (RTS/CTS) packet exchange between the sender and receiver. The sender transmits a very short RTS packet to the receiver, which replies by transmitting a similarly short CTS packet. The CTS tells nodes around the receiver about the following data packet, so that they can avoid interfering. Unfortunately, RTS/CTS cannot be used for broadcast packets, because there is no unique receiver specified to send back a CTS.

4.2.4 Asymmetry

If we build a radio link with two identical transmitters and receivers, we might think that the link would be symmetric. That is, the performance of the link, measured as throughput or percentage of packets received correctly, should be identical in each direction. Indeed, path loss and multipath effects are symmetric. However, receiver noise might be different at each end of the link. In addition, in a practical system, especially in a low-cost system, it is unlikely that the radios at each end of the link are precisely the same. For example, although both radios

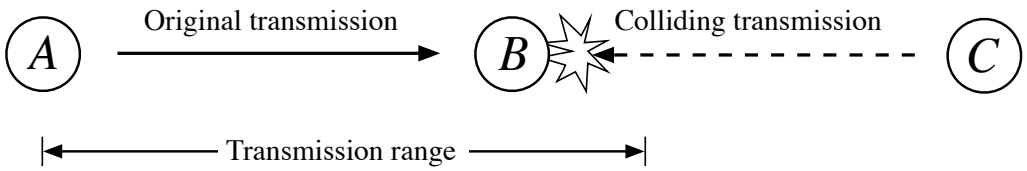


Figure 4-1: The hidden terminal problem. Because nodes *A* and *C* are out of range of each other, they are hidden terminals to each other, and neither can tell if the other is transmitting. As a result, they may transmit simultaneously, interfering at node *B*.

might be set to use the same transmit power, manufacturing and calibration differences and power supply differences (e.g. differences in battery level) can cause the transmit powers to be different.

4.3 Effect of Spread-Spectrum

Many modern packet radio systems use *spread-spectrum* techniques [63], which have numerous applications for radio and timing systems. This section briefly describes how spread-spectrum techniques can improve the performance of digital radios in the face of the narrow-band interference and multipath effects discussed in Section 4.2.

The basic idea behind spread-spectrum is that the signal transmitted by a radio is spread out over a much larger range of frequencies than necessary to convey the signal's information. For example, a signal that originally occupied 10 MHz is spread by a factor of ten to occupy 100 MHz. The receiver despreads the signal to its original frequency width before demodulating. Because the power of the signal is spread over a wider frequency range, the signal is less susceptible to interference that occurs in a narrow band of frequencies, such as that from other narrow-band transmitters. This interference only affects a fraction of the spread signal. The advantage that spreading gives over narrow-band interference is termed *processing gain*; as an example, 802.11 radios have about 10 dB of processing gain [18] at 1 Mbps. Spread-sprectrum can also help mitigate frequency-selective noise and path loss, by limiting their effects to a small fraction of the original signal. However, spread-spectrum does *not* provide any processing gain over white noise. Because white noise has the same power at all points in the spectrum, it affects a transmitted signal the same amount regardless of how widely the signal is spread.

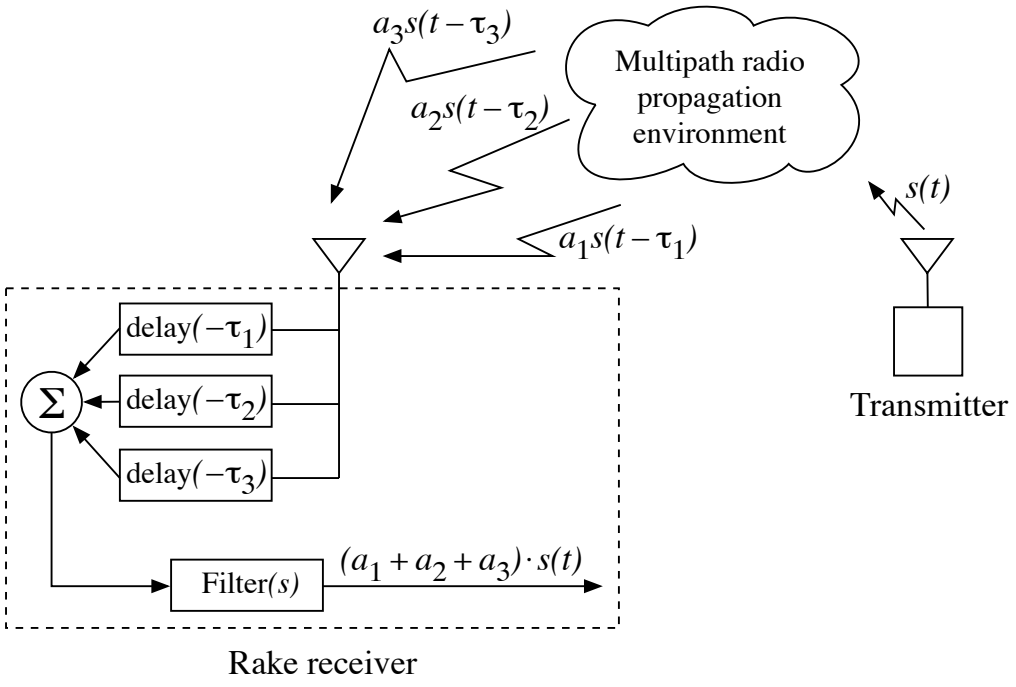


Figure 4-2: A Rake receiver. The radio propagation environment causes the the sum of three delayed and attenuated versions of the original signal $s(t)$ to arrive at the receiver. The Rake receiver splits the received signal into three copies, which it delays appropriately before adding them back together and filtering to recover an attenuated version of the original signal: $(a_1 + a_2 + a_3) \cdot s(t)$. The name of the Rake receiver comes from the structure of its multiple delay lines, which resembles a garden rake.

A second advantage of spread-spectrum is that it can be used to combat multipath effects, using a special sort of receiver design known as a *rake receiver* [64, 65]. The key feature of a rake receiver is that it is able to identify and compensate for the effects of multipath signals, either by filtering out delayed copies of the original signal, or by shifting them in time and recombining them into the original signal, as shown in Figure 4-2. Two important design parameters for a rake receiver are the number of delayed copies of the signal it can identify, and the range of delay for which it can compensate. These parameters can be chosen to match the receiver to the environment in which it operates. For example, the Intersil Prism 802.11 receiver chip, designed for indoor office wireless LAN applications, can handle up to 250 nanoseconds of delay spread at 5.5 Mbps, and 125 ns at 11 Mbps [35]. The measured delay spread of 2 GHz signals in an office building environment ranges from 50 to 150 nanoseconds [57, 58, 23].

4.4 Error Model

The characteristics of the RF channel discussed in Section 4.2 cause most wireless links to have some degree of error. Each link can be characterized in terms of its *loss ratio*, which describes what fraction of packets sent over the link will be incorrectly received. We use the term loss ratio because all damaged packets are treated as lost: any errors are detected using a checksum and discarded by the radio. This section describes a model for predicting the loss ratio at different packet sizes based on the measured loss ratios at a few known sizes. The experiments in Chapter 7 use spread-spectrum 802.11 radios, in the sort of indoor office environment for which they were designed. Therefore, we assume that the radios are robust to most narrow-band and multipath interference, and that errors are a result of a poor signal-to-noise-ratio (SNR) in an AWGN channel.

Before a receiver can correctly demodulate and decode a packet, the receiver must notice that the packet is being transmitted. The details of this depend on the receiver design, but typically the receiver will first notice a higher amount of RF power being transmitted on the frequency used by the radio link. The receiver will then try to synchronize with the signal by looking for framing information such as the preamble. Given that a packet has been transmitted over a particular link, the probability that the receiver successfully detects and synchronizes to that packet frame is P_f(SNR), which is a function solely of the wireless link's SNR (since we are assuming that all errors are due to poor SNR values). The exact form of P_f can be determined from the details of the radio design and implementation.

Once the receiver has detected that a packet is being transmitted, and the receiver is synchronized with the transmitter, the receiver can demodulate and decode each data symbol in the packet. The receiver may incorrectly demodulate a symbol; we again make the common assumption that any error is due to a poor SNR, and that the noise is AWGN. Under this assumption, symbol errors are independent, since the noise which causes any error is uncorrelated over time. Given that a receiver successfully detects and synchronizes to a packet over a particular link, we will write the per-symbol probability of each symbol in that packet being correctly demodulated as $P_s(\text{SNR})$. Like P_f, P_s is a function solely of the link's SNR, and the form of P_s depends on the details of the coding and modulation scheme being used.

For a packet with n data symbols, we can write the probability that all symbols are correctly received as P_s^n, since the probabilities of correctly receiving each symbol are independent. Therefore the probability of correctly receiving an entire packet is

$$P_p(\text{SNR}, n) = P_f(\text{SNR}) \times P_s(\text{SNR})^n \tag{4.1}$$

That is, P_p is the probability that the receiver detects and synchronizes to the packet, and successfully demodulates every data symbol in the packet. Since P_f and P_s are solely functions of the link's SNR, P_p is a function of the link's SNR and the packet size n. As described above, if we know all the relevant details of the receiver design and the radio's modulation scheme, we ought to be able to write out the function P_p. This is actually the link's *delivery ratio*, which is the complement of the link's loss ratio. Once P_p is determined, we can predict the delivery ratio of a link for a given packet size given that link's SNR. That is, if a link's SNR is measured to be s (perhaps using some statistics from the radio itself), we can calculate the delivery ratio for a packet of size n as $P_p(s, n)$.

However, determining the loss ratio of a link using P_p and the SNR is impractical, because the SNR information s can be hard to determine, and P_f may not be known. Some radios do not report accurate SNR information, or only report it for successfully received packets, biasing the SNR statistics. P_f may be unknown for several reasons. First, the design of the radio may be too complicated to model accurately; for some radio designs P_f is determined using Monte Carlo simulations [52]. Second, the detailed design of the radio may not be available for analysis to produce P_f. This is especially true if commodity radios are being used, as manufacturers are loath to give out the details of their hardware. Finally, although the per-symbol probability function P_s can be looked up from

a textbook for many modulation schemes (including those used by 802.11b), the radio's actual performance may differ from the theoretical performance by some margin. For example, the Intersil Prism 802.11 chipset has a measured symbol error performance of about 3 dB less than the theoretical performance at the 1 Mbps bit-rate [35]. The magnitude of this performance margin may not be known for a particular radio.

We sidestep these problems by measuring each link to to determine its P_f and P_s. We assume that each link has some fixed, but unknown SNR, at least over a period of time long enough to take measurements. Then P_f and P_s are fixed but unknown quantities for each link. By measuring P_p at two known packet sizes n_1 and n_2, and using equation 4.1, we end up with two equations which can be solved for the two unknowns, P_f and P_s:

$$P_p(n_1) = P_f \times P_s^{n_1} \tag{4.2}$$
$$P_p(n_2) = P_f \times P_s^{n_2}$$

Let $R = P_p(n_1)/P_p(n_2)$, $\Delta = n_1 - n_2$, and assume that both packet sizes had non-zero probabilities of being successfully received. Then

$$P_s^{\Delta} = R \tag{4.3}$$
$$\Delta \ln P_s = \ln R \tag{4.4}$$
$$P_s = e^{\frac{\ln R}{\Delta}} \tag{4.5}$$

Substituting equation 4.5 into equation 4.2 gives

$$P_f = \frac{P_p(n_1)}{P_s^{n_1}} \tag{4.6}$$
$$= \frac{P_p(n_1)}{e^{\frac{n_1 \ln R}{\Delta}}} \tag{4.7}$$

4.4.1 Model Inaccuracies

The loss model presented above is extremely simple. For example, it assumes that each link's SNR does not change over time, or if it does, that it changes slowly enough that we can get accurate and consistent measurements for P_f and P_s. Also, like Modiano [54], this model assumes that each symbol error is independent. In

general, channel noise and interfering transmissions are not AWGN, and will be correlated in time: a symbol is more likely to be received in error if the previous symbol also encountered an error. The model doesn't account for coding techniques such as forward error correction (FEC), which allow a receiver to correctly reconstruct the data in a packet despite some number of errors occurring. The details of how many symbol errors can be tolerated depend on how many errors there are, where the errors are in relation to one another, and how many bits are affected by each symbol error. Accounting for all of these details, even assuming AWGN, requires relatively complex analysis that is beyond the scope of this chapter.

Despite the model's simplicity, it is consistent with some previous network measurement results. For example, Nguyen et al. [56] report indoor loss and error measurements of the AT&T WaveLAN, a 900 MHz spread-spectrum radio. Their results show that packet delivery ratios decrease exponentially with increasing packet size. Duchamp and Reynolds [27] also report the results of indoor experiments with the WaveLAN radio, concluding that the average number of errors per bit in received packets is independent of packet size. Although this result does not imply that bit or symbol errors are independent, it is consistent with that assumption. Willig et al. [77] present error measurements of a radio implementing the IEEE 802.11 physical layer at 2.4 GHz. Their results show that although bit errors are highly correlated, the number of errors per bit does not seem to depend on the packet size, as in the 900 MHz WaveLAN measurements. The next section shows that although the symbol independence assumptions used to derive the loss model may be too strong, the model still provides accurate delivery ratio predictions.

4.4.2 Model Evaluation

To determine the accuracy of the model's loss ratio predictions, seven sets of broadcast experiments were run over two days. During each experiment, for each one-hop link, the source node sent broadcast packets of eighteen different Ethernet sizes, from 50 to 1,500 bytes. Packets were sent at the 1 Mbps bit-rate. The destination node of each link recorded how many packets it received of each size from each sender. The delivery ratio of each packet size over each link is calculated as the number of packets of that size received over the link divided by the number of packets of that size that were actually sent over the link. To smooth out variability over time, the results from each of the seven experiments are averaged, resulting in a single delivery ratio for each one-hop link and packet size.[1]

[1] These experiments were performed by Daniel Aguayo.

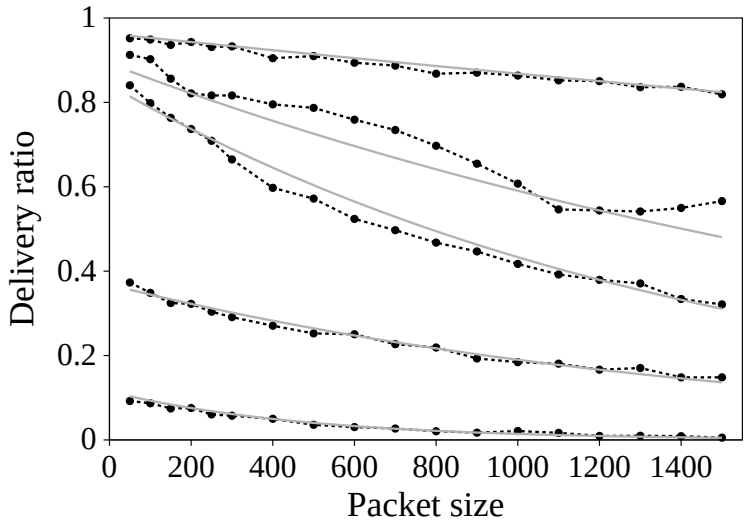


Figure 4-3: Example predicted loss ratios for a few links, using the model from Equation 4.1. The parameters P_f and P_s were calculated using the measured loss ratios of 200- and 1,200-byte packets, which give the best overall results, as shown in Figure 4-5. The smooth gray lines show the predicted delivery ratios for each link, and overlay the dashed black lines with points which show the measured delivery ratios for the same link at various packet sizes. Data are shown for five links: 12 → 1, 11 → 13, 23 → 36, 21 → 19, and 17 → 18. The measured packet sizes are 50, 100, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1,000, 1,100, 1,200, 1,300, 1,400, and 1,500 bytes.

Figure 4-3 shows the measured loss ratios at each size for a few example links. The figure illustrates how packet delivery ratios decrease with increasing packet size. The figure also shows the predicted delivery ratios, calculated using Equation 4.1, superimposed over the measured delivery ratios for each link. Although a few links do not have a perfect exponential delivery ratio structure, in general the model closely matches the measured delivery ratios.

Figure 4-4 shows the loss ratio prediction accuracy of Equation 4.1 for each packet size shown in Figure 4-3, and all links, using the measured loss ratios of 200- and 1,200-byte packets. For each size and link, a tiny circle shows the predicted delivery ratio of that size over the given link, versus the measured delivery ratio of that link. Most points lie very close to the line $y = x$, indicating that overall, the model is very accurate. The median prediction error is 0.006.

Figure 4-5 shows how the model performs when we vary the size of the packets whose loss ratios are measured. Each line shows the delivery ratio prediction performance using measurements at a given pair of packet sizes. Each point on a line shows the average prediction error at that size across all links. Clearly, the model more accurately predicts delivery ratios for sizes closest to the sizes whose delivery ratios were actually measured. Using two small packet sizes will lead to inaccurate predictions for large packets; similarly, using two large packet sizes will give poor predictions for small packets. Using one small and one large packet size gives the smallest average error, as this spreads the errors out across all packet sizes. However, the best choice of which sizes to measure at will ultimately depend on the packet sizes we are interested in.

The distribution of prediction errors by size is particularly relevant for the route metric calculation, discussed in Chapter 5, which depends on the delivery ratio of 802.11 ACK packets. We would like to know how accurately we can predict the delivery ratio of ACK packets over a link. Unfortunately, we cannot directly evaluate the prediction accuracy for ACK packets, primarily because with the 802.11 protocol there is no way to measure the delivery ratio of ACK packets in isolation. It is impossible generate ACK-sized packets without first generating a much larger data packet in the reverse direction. Unlike data packets, whose loss ratio we can measure directly by sending broadcasts, the transmission of an ACK packet is conditional on the successful reception of the data packet.

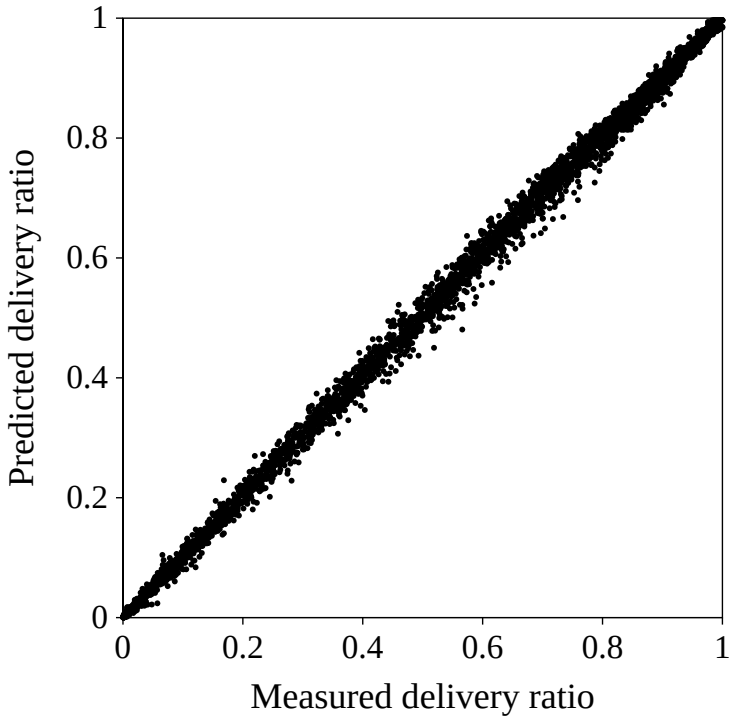


Figure 4-4: Scatter plot of predicted delivery ratio versus measured delivery ratios. Each point shows the predicted versus measured delivery ratios for one link and one packet size. The predicted delivery ratio for each point is calculated using Equation 4.1 and the measured delivery ratios of 200- and 1,200-byte packets, as in Figure 4-3. The packets sizes used are the same as in Figure 4-3, except for 200- and 1,200-byte packets, which were used for the curve fit (200 and 1,200 bytes). 22 of 339 links were omitted because 1,200-byte packets had a higher delivery ratio than 200-byte packets. The graph has 5,062 points.

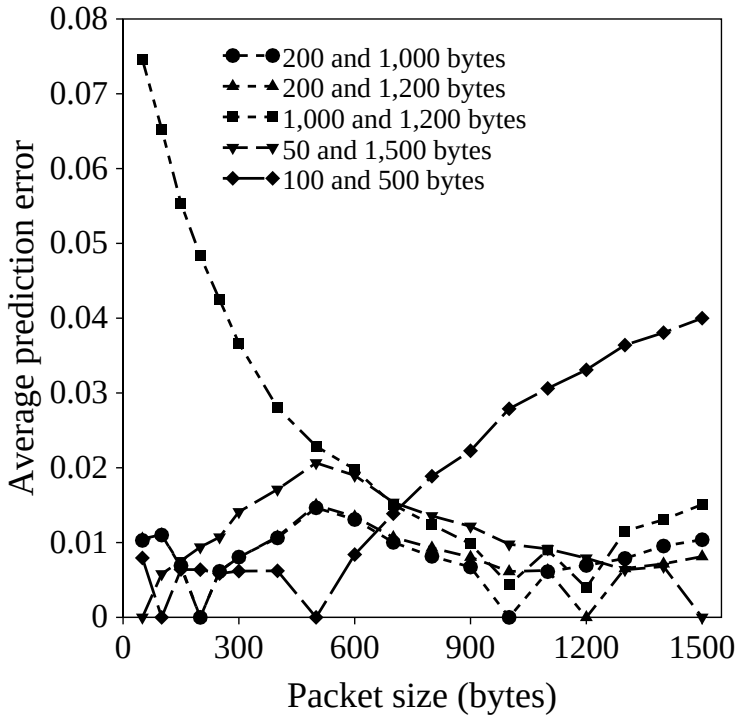


Figure 4-5: Distribution of delivery ratio prediction errors by packet size, using the two-size exponential model in Equation 4.1. Each line shows the prediction errors at various packet sizes for a given pair of sizes used to find P_f and P_s. Each line touches the x axis at the packet sizes used to make predictions; the predictions become worse as the sizes move further away from those used for the prediction. The best average error across all sizes and links is obtained using the measured delivery ratios of 200- and 1,200-byte packets.

Chapter 5

Design of ETX

This chapter describes the design of the expected transmission count (ETX) metric for finding high-throughput routes. The ETX design is motivated by the causes of low throughput described in in Chapter 3: lossy and asymmetric links, and contention between links in a route. This chapter also includes a discussion of why ETX is better than some other proposed metrics.

5.1 ETX Intuition

The goal of ETX is to find high-throughput routes. The main intuition behind the ETX design is that because links in a route share the wireless spectrum, protocols can increase throughput in packets per second by decreasing the amount of time each packet uses that spectrum. One way to do this is for protocols to choose routes with fewer links, that is, find minimum hop-count routes.

However, as Chapter 3 showed, a minimum hop-count route may not be the highest-throughput route, if it uses lossy links. Since the 802.11 protocol uses link-level retransmissions, it takes more time to send a packet over a lossy link. This time reduces route throughput in the same way that adding links to a route reduces route throughput: while the sender is retransmitting packets over a lossy link, other links in the route are unable to send. So, in addition to using shorter routes, protocols should also try to use less lossy links.

The second intuition behind ETX is that these two criteria can be combined into one: the extra transmissions due to adding links can be lumped with the retransmissions on lossy links, producing a total number of transmissions for a path. Protocols should find routes that reduce that total number of transmissions per

packet. Routes with fewer total transmissions per packet have higher throughput, because they take less time to send a packet.

5.2 Design Criteria

The goal of ETX is to find high-throughput routes by choosing routes with the fewest transmissions per packet. Chapter 3 described several aspects of link behavior that affect route throughput:

Broad Distribution of Link Loss Ratios The distribution of link loss ratios is relatively evenly spread from very lossy links to very good links, as shown in Figure 3-5. As a result, the metric should avoid discarding links based on loss ratios. This is primarily because even a lossy link may provide higher throughout over a one-hop route than any available multi-hop routes. It would also be hard to select which threshold should be used classify links. For any reasonable threshold, there are likely to be many links which could be useful, but whose loss ratios are slightly greater than the threshold.

Asymmetric Loss Ratios The loss ratios in both directions of a link are often different. For example, a link may deliver all of its data packets in on direction, but drop most of the 802.11 acknowledgment packets in the reverse direction. The metric should consider loss ratios in both directions, and should not draw conclusions about one direction of a link based on its performance in the other direction.

Multi-hop Interference As described in Chapter 3 and Li et al. [47], successive hops of a route interfere with each other, reducing throughput even when all links successfully deliver every packet. The metric should account for this intra-route interference as well as the effects of lossy links.

5.3 The ETX Metric

The ETX metric for a link is the expected number of data transmissions required to send a packet over the link, including retransmissions. The ETX metric of a route is the sum of the ETX metrics for each link in the route. For example, the

ETX of a three-hop route with perfect links is three; the ETX of a one-hop route with a 50% delivery ratio is two.

The ETX of a link is calculated using the forward and reverse delivery ratios of the link. The forward delivery ratio, d_f, is the measured probability that a data packet successfully arrives at the recipient; the reverse delivery ratio, d_r, is the probability that the ACK packet is successfully received by the data sender, given that the data packet was received successfully. The probability that a data transmission is successfully received and acknowledged is $d_f \times d_r$. A sender will retransmit any data packet that is not successfully acknowledged. Because each attempt to transmit a packet can be considered a Bernoulli trial, the expected number of transmissions for a link is approximated[1] as:

$$\text{ETX} = \frac{1}{d_f \times d_r} \tag{5.1}$$

This equation assumes that the probabilities d_f and d_r are constant for a given link, or are at least constant for the duration of link measurements.

ETX has several important characteristics:

- ETX is based on packet delivery ratios, which directly affect throughput.
- ETX detects and appropriately handles asymmetry by incorporating loss ratios in each direction.
- ETX can use precise link loss ratio measurements to make fine-grained decisions between routes.
- ETX penalizes routes with more hops, which have lower throughput due to interference between different hops of the same path [47].
- By minimizing transmission counts, ETX tends to minimize spectrum use, which should maximize overall system capacity.

In addition, ETX may decrease the energy consumed per packet, as each transmission or retransmission may increase a node's energy consumption.

The delivery ratios d_f and d_r are measured using dedicated link probe packets. Each node broadcasts link probes of a fixed size, at an average period τ (one second in the implementation). To avoid accidental synchronization, τ is jittered by

[1] Since real hardware limits the number of maximum retransmissions per packet, the actual number of retransmissions per packet will be slightly less.

up to ±10% per probe. Because the probes are broadcast, they are not acknowledges or retransmitted. Every node remembers the probes it receives during the last w seconds (ten seconds in our implementation), allowing it to calculate the delivery ratio from the sender at any time t as:

$$d(t) = \frac{\text{count}(t - w, t)}{w/\tau}$$

Count$(t - w, t)$ is the number of probes received during the window w, and w/τ is the number of probes that should have been received. In the case of the link $X \rightarrow Y$, this technique allows X to measure d_r, and Y to measure d_f. Because Y knows it should receive a probe from X every τ seconds, Y can correctly calculate the current loss ratio even if no probes arrive from X.

Calculating a link's ETX requires both d_f and d_r. Each probe sent by a node X contains the number of probe packets received by X from each of its neighbors during the last w seconds. This allows each neighbor to calculate its d_f to X whenever it receives a probe from X.

The ETX of a route R is the sum of the link metrics for each link l:

$$\text{ETX}(R) = \sum_{l \in R} \text{ETX}(l) \tag{5.2}$$

$$= \sum_{l \in R} \frac{1}{d_f^l \times d_r^l} \tag{5.3}$$

where d_f^l and d_r^l are the forward and reverse delivery ratios for each link l in the route R. DSDV accumulates this metric sum as it forwards route updates. DSR can accumulate the metric sum as it forwards queries, or at the querying host once all route replies have been received.

If the highest-throughput path has three or fewer hops, ETX is likely to choose it: the throughput of these paths is determined by the total number of transmissions, since all of the hops interfere with each other [47]. If the best path has four or more hops, ETX may choose a slower path with fewer hops, since the extra transmissions required by extra hops do not slow down throughput beyond three hops. Route throughput is also affected by the amount of back-off at each node. Two links with the same average ETX but different patterns of packet loss over time can have different throughputs, which ETX does not account for.

5.3.1 ETX Assumptions

ETX makes several assumptions about the link layer. First, ETX is designed for networks with per-link retransmissions, like 802.11 provides. Networks with end-to-end retransmissions will have a different expression for the number of transmissions per packet.[2]

Second, ETX assumes that radios have a fixed transmit power level. With variable power radios, it might be preferable to maximize hop-count, thereby decreasing interference and minimizing the energy used by each packet [68, 32, 40].

Third, ETX assumes that each node has a single half-duplex radio, and that no two links can use the same radio spectrum simultaneously. If successive links transmit on different logical or physical radio channels, or if nodes have multiple radios, it may be possible for every link in a route to send packets at the same time.

Finally, ETX assumes that all links operate at the same bit-rate. However, when links can run at different bit-rates, it might be faster to use a lossy high bit-rate link than a perfect low bit-rate link.

ETX does not attempt to route around congested links, and in theory should not suffer from the oscillations that sometimes plague load-adaptive routing metrics such as end-to-end delay [9, 42]. To a first approximation, the loss measurements used by ETX do not reflect how busy a link is; a busy link may cause a probe broadcast to be deferred, but won't ordinarily cause it to be lost. This is not true, however, when the network is subject to heavy load like the UDP streaming used in Chapter 7. Because RTS/CTS cannot be used for broadcasts, the 802.11 probe broadcasts are vulnerable to collisions from hidden terminals. Also, 802.11 MAC unfairness can prevent probes from being sent on time. Because the most recent successful sender always resets its back-off window to the minimum size, it is most likely to succeed in the next contention window [8, 10, 55]. To address this unfairness in the 802.11 MAC, senders that do not get a chance to transmit will continuously decrement their back-off timers during any idle time, including during other senders' back-off. This ensures that eventually one of the nodes waiting to send will win back-off, send a packet, and continue to win back-off for a while. However, even though a node receiving heavy data traffic will eventually get to win back-off and send a probe, that probe (and following probes) could be substantially delayed, reducing the number of probes sent during a given time window. As a result, the node's neighbors will believe that the reverse delivery

[2] The actual expression will depend on the details of the retransmission policy, and, unlike ETX, is likely to depend on the order of the links in the route.

ratios are very small, and calculate a large metric, causing the routing protocol to avoid using the link.

ETX does not specifically account for mobility. ETX may choose good paths despite mobility if the underlying routing protocol can propagate route metrics quickly enough, and if accurate link measurements are available. One way to quickly obtain good ETX estimates might be to use the number of retransmissions per packet reported by the 802.11 interface, but these metrics would still need to be propagated around the network. In general, there is a trade-off between the accuracy of link measurements and a routing protocol's responsiveness to mobility.

5.4 Alternative Metric Designs

There are many other techniques and routing metrics proposed for finding high-throughput paths in multi-hop wireless networks. This section discusses some of those techniques with an eye to the design criteria in Section 5.2.

Masking Errors A simple approach to handling loss links is to mask transmission errors, either with retransmissions or error correcting codes. For example, the 802.11 ACK mechanism resends lost packets, making all but the worst 802.11 links appear loss-free. However, retransmission and coding do not make lossy links more desirable for use in routes, as they simply convert lossy links into slow links. The routing protocol should instead find links with lower loss ratios.

Thresholds Minimum hop-count routing could be augmented by ignoring links with loss ratios above a specified threshold. However, a below-threshold link may be the best way to reach some node. Also, there may be significant loss ratio differences even among the above-threshold links.

End-to-End Delivery Ratio The end-to-end delivery ratio of a route is the product of the per-link delivery ratios along that route; protocols choose routes with the highest product. This metric fails to account for inter-hop interference; it would view a perfect two-hop route as better than a one-hop route with a 10% loss ratio, when in fact the one-hop route would have almost twice the throughput.

Bottleneck Bandwidth Protocols choose routes with the highest bottleneck link throughputs. Although this approach may work well for networks where links are relatively isolated from each other, such as wired networks or wireless networks with directional links, it doesn't account for the inter-hop contention in wireless networks with omnidirectional antennas. For example, the bottleneck metric would consider a one-hop route with a 10% loss ratio to be equivalent to a two-hop route consisting of two links, each with a 10% loss ratio. However, the one-hop route would actually have about twice the throughput.

End-to-End Delay End-to-end delay is influenced by several factors, including the number of transmissions along a route, queuing time, and MAC protocol back-off time. All else being equal, it is probably desirable to choose a route that decreases the delay due to any of these factors. However, end-to-end delay changes with network load as interface queue lengths vary, while the goal of this work is to design a metric that is independent of network load. Load balancing and traffic engineering to decrease queuing and back-off times can be performed with separate algorithms.

Signal Strength Many radios can provide measurements of the received signal strength for each link. In theory, link signal strength should predict the probability of packet errors on that link. Figure 5-1 shows the measured relationship between short-term packet delivery ratios and the received signal strength reported by the radios for a few links in the test-bed network. In practice, using signal strength does not seem to be a practical approach for commodity 802.11 hardware, as there is no good relationship between the signal strength and delivery ratios. Since signal strength measurements are only reported by the radio for successfully received packets, the data is biased. Also, in addition to low signal-to-noise ratios, packet errors can be caused by multipath effects, which are not captured by the receiver's signal measurements.

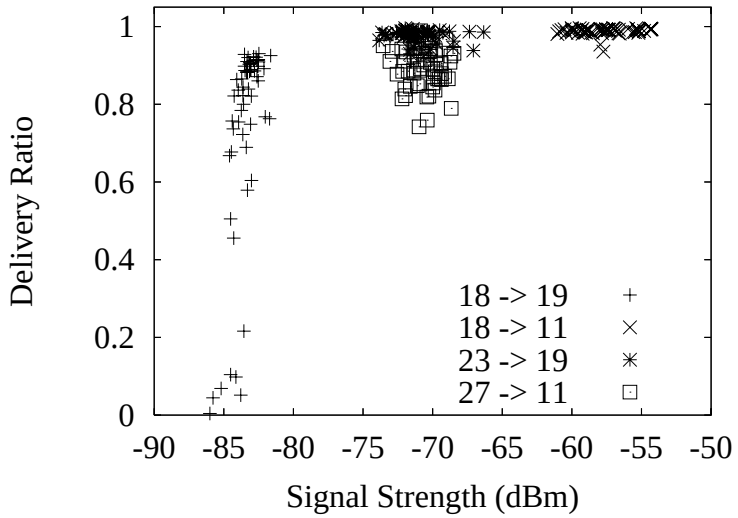


Figure 5-1: Received signal strength does not predict packet delivery ratios. This figure shows delivery ratios measured over 1 second versus the average received signal strength during that time, for four sample links.

Chapter 6

Protocol Implementation

The routing system in which ETX is implemented has four main parts: the Click toolkit [44], Click-based implementations of the DSDV [61] and DSR [37] routing protocols, and the ETX link measurement algorithm. This chapter describes the implementation details of DSDV, DSR, and ETX, as well as the route metric abstraction that enables these DSR and DSDV implementations to work with many different route metrics.

Because the Click toolkit can run in both user-space and in the kernel, so can the protocol implementations described here. However, running in the kernel provides a few important implementation advantages, such as priority queuing, and easy access to transmission failure notification from the 802.11 MAC layer.

The DSDV protocol is implemented following the description by Perkins and Bhagwat [61], with ambiguities resolved by consulting Broch et al. [11] and the Rice/CMU implementation in the *ns* simulator [59, 66]. The DSR implementation follows the IETF Internet-Draft, version 9 [38].

6.1 Operation of DSDV

DSDV is a distance-vector protocol, which uses sequence numbers to ensure freshness, and a *settling time* mechanism to avoid unnecessarily propagating any routes with inferior metrics. We made four changes to the original DSDV design in order to ensure that it uses the path with the best known metric. Before describing those changes, we present an overview of how the published version of the protocol selects routes.

Every node has a routing table entry for each destination *D*, which contains four fields: *D*'s identifier (IP address), the next hop on the route to *D*, the latest sequence number heard for *D*, and the route metric. A node forwards packets to the next hop specified by the current contents of its routing table.

Every node periodically broadcasts a route advertisement packet containing its complete routing table. This advertisement is known as a *full dump*, and occurs at the *full dump period*. Each node maintains a sequence number for itself, which it increments and includes in its own entry in every full dump it originates. The node copies the sequence numbers for the other entries in the full dump from its routing table. The effect is that the sequence number field in a routing table entry or advertisement entry reflects the age of that entry's routing information.

When a node receives another node's route advertisement broadcast, it updates its own route entries as follows. Suppose node *X* receives an advertisement from *Y* for destination *D* with metric *m* and sequence number *n*. If *n* is newer than the sequence number in *X*'s current entry for *D*, *X* replaces its current entry with the new route through *Y*. *X* also accepts the new route if the sequence number is the same, but *m* is better than the metric of the current route. If *X* has no route to *D*, it accepts the new route. Otherwise *X* ignores the advertised route.

Each route entry has an associated *weighted settling time* (WST). The settling time of a route entry with a given sequence number is the amount of time between when a route with the sequence number was first received, and the time when the best route with the same sequence number was received. The WST is the weighted average of the settling times for recent sequence numbers, and is updated whenever a route with a new sequence number is received.

The WST is used together with *triggered updates* to quickly propagate good routes through the network, while avoiding an explosion of broadcasts. Whenever a node replaces a route entry with a newly received entry, it propagates the new route to its neighbors by sending a triggered update which contains only the changed information. However, triggered updates are not sent until at least 2×WST has passed since first hearing the current sequence number. This prevents nodes from advertising a new route which will likely be replaced later by a better route. In addition, regardless of each route entry's WST, triggered updates are sent at no more than a maximum specified rate. Triggered updates that are delayed are batched together and sent at the next available time.

Finally, DSDV specifies that triggered updates can become full dumps if a large enough fraction of the routes need a triggered update. In this case, all routes with an elapsed WST are included in the full dump, and the node's sequence number is incremented.

6.2 Changes to DSDV

The DSDV algorithm we implemented differs from the the CMU *ns* DSDV implementation in four ways that improve its performance in the test-bed network. The first two changes were made based on observations from the literature, while the third and fourth changes were motivated by pathological DSDV behavior observed on the indoor network using detailed packet traces.

The first change affects how the WST is used. The *ns* DSDV implementation does not advertise a route entry until 2×WST has passed since that *particular* route entry to the destination was heard. However, according to our interpretation of the original DSDV description [61], the waiting time before advertising a route should start when the *first* route of each sequence number is heard. Because each node's WST is an estimate of the time between when the node first hears a given sequence number for a destination and when the node hears the best metric with the same sequence number for that destination, the node assumes that it has the best route for a given sequence number after 2×WST has passed. Then it is likely that no better route will be heard for that sequence number, and the best route heard so far should be propagated.

The second change is that our implementation does not use link-level feedback (i.e. 802.11 transmission failure notices) to detect broken links and produce broken-route advertisements. Broch et al. [11] report that broken-route advertisements due to link-level feedback typically cause all routes to the particular destination to be broken throughout the whole network, not just those that use the broken link. This makes the destination effectively unreachable from anywhere until its next route advertisement. Our implementation still generates broken-route advertisements when routing table entries time out, but this rarely occurs during the experiments.

The third change is that full dumps are never sent on a triggered update, even if many routes have changed. Triggered updates contain only the changed routes, and full dumps are only sent at the full dump period. This change significantly decreases the routing protocol overhead on our network. Because the indoor test-bed is dense, each node exchanges advertisements with a large fraction of the network's nodes. If a full dump were sent on a triggered update, the sender's new sequence number would in turn trigger a cascade of triggered updates from its neighbors, increasing the amount of protocol overhead.

The fourth and final change (called *delay-use*) is that a route is not used until it is allowed to be advertised. That is, a new route is not used until 2×WST has expired since its sequence number was first heard. With this change, the best route

heard for the previous sequence number is used until the current sequence number's WST has expired. Unmodified DSDV always uses the latest route accepted for a given destination, even if it cannot yet advertise that route.

Delay-use prevents DSDV from prematurely using routes with bad metrics. For example, if there is an asymmetric one-hop route, a node will always hear new sequence numbers along the one-hop link first. Without delay-use, DSDV is forced to immediately use the new one-hop route for routing, even if the ETX metric is poor. In general, shorter routes deliver new sequence numbers first, causing the original DSDV to use shortest paths for some fraction of the time between successive sequence numbers, regardless of the metric in use. With delay-use, DSDV will use the best route with the previous sequence number until the WST has expired and the best route with the new sequence number has likely been heard. Section 7.1.2 shows that delay-use improves the performance of DSDV with ETX.

Figure 6-1 shows pseudo-code for the DSDV routing table update and packet forwarding algorithms, including our changes. The full dump period was 15 seconds, and routing table entries were timed out after 60 seconds. Triggered updates were issued at a maximum rate of one per second. All the DSDV experiments used the four protocol changes described above, unless otherwise noted.

The ETX implementation measures link loss ratios with small probe packets, as described in Chapter 5. Probes contain 134 bytes of 802.11 payload. An ETX node broadcasts one probe per second, and remembers probes received from neighbors over the last ten seconds. Using relatively small probes saves bandwidth; Chapter 7 shows that predictions based on small packets are still useful even when the data traffic consists of large packets.

6.3 DSR Implementation

Our DSR implementation follows revision 9 of the IETF Internet-Draft specification [38], following the requirements for networks which require bidirectional links to send unicast data. The implementation is derived from Click-based DSR implementations originally developed at the University of Colorado at Boulder [24, 75]. This section reviews DSR's basic operation as described in the draft, and describes our modifications to support ETX and other metrics.[1]

DSR is a reactive routing protocol, in which a node issues a *route request* only when it has data to send. Route requests are flooded through the network,

[1]The DSR implementation was written by Daniel Aguayo.

```
handle_route_ad(Packet p) {
 foreach Route r in p do
   handle_update(r);
}

handle_update(Route r) {
  // curr[]: best route for current seq
  // old[]: best route for previous seq

  // add link-metric to r.metric
 update_metric(r);

 if (r.seq == curr[r.dest].seq && r.metric < curr[r.dest].metric) {
   curr[r.dest] = r;
   curr[r.dest].best_time = now;
   schedule_triggered_update(r);
 }
  else if (r.seq > curr[r.dest].seq) {
    // save best route of last seq no
   old[r.dest] = curr[r.dest];

   curr[r.dest] = r;
   curr[r.dest].first_time = now;
   curr[r.dest].best_time = now;

    // update settling time
   old_wst = old[r.dest].wst;
   best_t = old[r.dest].best_time;
   first_t = old[r.dest].first_time;
   curr[r.dest].wst = 0.88×old_wst + 0.12×(best_t – first_t);

   schedule_triggered_update(r);
 }
  // ignore old seqnos and bad metrics
}

// returns next hop ip address for dst
lookup_route(IPAddress dst) {
  // use old route if we haven't yet advertised current route
 if (curr[dst].first_time + 2×curr[dst].wst > now)
   return old[dst].next_hop;
 else
   return curr[dst].next_hop;
}
```

Figure 6-1: DSDV pseudo-code, including the modifications described in Section 6.2. The WST parameters 0.12 and 0.88 are chosen to produce a reasonable average.

each node appending its own address to each request it receives, and then rebroadcasting it. Each new request includes a unique ID, which forwarders use to ensure they only forward each request once.

The request originator issues new requests for the same destination after an exponentially increasing back-off time. Route requests are issued with increasing time-to-live (TTL) values, to minimize the range and cost of flooding.

The destination issues a *route reply* in response to every forwarded request it receives. Each reply, which includes the route which was accumulated as the request was forwarded through the network, is source-routed back to the originator along the reverse route. The source node chooses a route using information from the route replies it receives, and source-routes data along this route.

Our implementation stores the results of route replies in a *link cache*, which stores information about each link separately. A node runs Dijkstra's shortest-path algorithm on its link cache to find the best route to a destination.

DSR uses feedback from the link layer to react to link failures. When the 802.11 card signals that no acknowledgment was received after the maximum number of retries, the forwarding node issues a *route error* back to the source, which removes the link from its link cache and then computes a new route. If the source cannot find a route using its link cache, it issues a new route request.

To deal with asymmetric links, each node maintains a *blacklist*, which lists immediate neighbors with unidirectional links to the the node. These are links over which the node might receive broadcast requests, but which are unsuitable for unicast traffic. If a transmission failure occurs when forwarding a route reply, the neighbor to which the node was trying to forward the reply is added to the blacklist, with an entry of *unidirectionality probable*. From that point, the node will not forward route requests received over that link. If the asymmetry of the link is not positively determined for some time, its entry is downgraded to *unidirectionality questionable*. If a route request is received over such a link, the node delays forwarding it while it issues a direct, one-hop unicast route request back to the questionable neighbor. If a reply is received, the node forwards the original route request and removes the blacklist entry. Otherwise, the node drops the request. Entries are removed from the blacklist when the link is determined to be bidirectional, e.g. by a successful unicast transmission.

The DSR specification describes optimizations in which nodes update their link caches using data from packets they forward or overhear on the air. We did not implement any of the optimizations which require the wireless interface to operate in promiscuous receive mode. We also did not implement 'reply from cache,' in which forwarding nodes can respond to route requests with information from their own link caches. All link caches were flushed between experiments, so these decisions should not affect the results in this work.

The nodes do not perform packet salvage, where forwarding nodes try to find alternate routes for queued packets when the head-of-queue packet has a transmission failure, or a route error is received. Instead, queued packets with invalid route information are simply removed from the queue and dropped. Because this implementation has only a five-packet queue, a maximum of five packets could be salvaged at each error, which would not increase throughput appreciably.

To use ETX and other metrics besides hopcount, the implementation was modified in a few simple ways. Link probes are used to measure delivery ratios, as in the DSDV implementation. When a node forwards a request, it appends not only its own address, but also the metric for the link over which it received the request. These metrics are included in the route replies sent back to the sender. When a node receives a request which it has already forwarded, it forwards the request again if the accumulated route metric is better than the best which it has already forwarded with the same ID. This increases the chances that the originator will hear about the route with the best metric.

Entries in the link cache are weighted by the metrics which were included in the route replies. The Dijkstra algorithm finds the route to the destination which has the minimum metric.

6.4 Router Configuration Details

If a node is sending large volumes of data, there is a danger that probe packets or routing protocol packets may be dropped or delayed due to a full queue. To mitigate this problem, the implementation maintains separate Click queues for data packets, protocol packets, and link probes. Each of these queues can hold five packets. These queues all drain into a single queue in the wireless adapter's memory, managed by the driver, which has a capacity of three packets. Loss-ratio probes enter the adapter's queue first, followed by protocol packets, then data packets.

The DSDV implementation looks up a packet's destination in the routing table after dequeuing the packet from the data queue, and just before handing the packet to the 802.11b card. This avoids committing to the next hop before queuing, and makes forwarding more responsive to changes in the routing table. This technique depends on the fact that the nodes have only one wireless interface. Figure 6-2 shows the DSDV queuing configuration.

The DSR implementation, on the other hand, adds the source-route header to data packets before inserting them into the queue. On a transmission failure or a

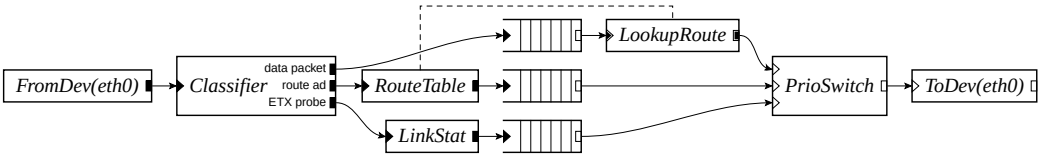


Figure 6-2: DSDV queuing configuration. Incoming packets are classified as data, route broadcasts, or ETX probes. Data packets are immediately queued; *LookupRoute* determines each data packet's next hop by looking in the *RouteTable* when the packet is dequeued and sent to the interface. Route broadcasts are sent to the *RouteTable*, while ETX broadcast probes are sent to the *LinkStat* module, which maintains link delivery ratio estimates. *RouteTable* periodically produces new route advertisements for the local node, and *LinkStat* periodically produces new ETX probes; these are enqueued separately. *PrioSwitch* pulls packets from the queues in priority order, sending them to interface. It give priority to ETX probes, then route broadcasts, then data packets.

received route error, a node removes and drops all enqueued packets which include the broken link in their source route. This ensures that the node experiencing the transmission failure does not spend additional time and spectrum retransmitting more packets over the broken hop.

6.5 Modular Route Metrics

The protocol implementations take advantage of Click's modularity to use modular route metric implementations. That, is the metric implementations are not part of the protocol implementations, but are instead modules that the protocol implementations are configured with at run time. This increases the maintainability of the protocol implementations by allowing them to work with new route metrics without modifying the protocol code, and allows different protocols to share the same metric implementations, reducing the work to test a new metric with different protocols.

The drawback of modular route metrics is that the metrics and protocols must use a fixed generic metric abstraction which may not map well to a particular metric or protocol. In practice the benefits outweigh the drawbacks. For example, the DSR and DSDV implementations can share at least eight different metric modules, despite the different structure of the protocols.

```
// Abstract metric datatype
struct metric_t {
  bool valid;          // Is this metric valid?
  int    metric_val;   // Actual metric data; opaque to protocol code
}

// Is the abstract value of M1 'better' than M2? (M1 < M2)
bool  metric_val_lt(metric_t M1, metric_t M2)

// Return the link metric from this node to neighbor N.  DATA_SENDER?
// is true if this node is sending data to N over the link, false if N is
// sending data to this node over the link.
metric_t  get_link_metric(LinkAddress N, bool DATA_SENDER?)

// Return the metric for the route formed by appending the link with
// metric L to the end of the route with metric R.
metric_t  append_metric(metric_t R, metric_t L)

// Return the metric for the route formed by prepending the link with
// metric L to the front of the route with metric R.
metric_t  prepend_metric(metric_t R, metric_t L)
```

Figure 6-3: Generic metric abstraction. This interface is compatible with the structure of both the DSDV and DSR protocols.

```
// Return measured delivery ratio for sending packets to neighbor N,
// as 0-100 percent
int  get_forward_ratio(LinkAddress N)

// Return measured delivery ratio for receiving packets from neighbor N,
// as 0-100 percent
int  get_reverse_ratio(LinkAddress N)
```

Figure 6-4: Link measurement interface. All metric implementations that require link delivery ratio measurements use this interface.

The generic metric abstraction is shown in Figure 6-3. A distributed routing protocol can calculate metrics incrementally at each node as it build routes one link at time. When each node adds a new link to a route it is building, the protocol first calls get_link_metric(), then append_metric() (or prepend_metric() as appropriate) to combine the new link's metric with the route metric so far. In a centralized protocol, each node can call get_link_metric() for each of its links, then send those link metrics to a central node for route computation; that central node will repeatedly call append_metric() to calculate the metric for each route. This route metric abstraction only works for route metrics than can be calculated incrementally.

The value of the DATA_SENDER? flag passed to get_link_metric() specifies whether or not the node calculating the link metric will be sending or receiving data over the link. This distinction is important because not all metrics are symmetrical, and different routing protocols calculate the link metric at different ends of a link. For example, DSDV calculates link metrics at the data sender end of each link, as the route advertisements flood through the network away from the destination. DSR calculates link metrics at the data recipient end of each link, as route requests flood through the network away from the data sender.

Another modular feature of the protocol implementations is that the link delivery ratio measurement code is contained in its own module that is also configured at runtime. Since many of the metric modules need to know link delivery ratios (such as ETX, bottleneck loss ratio, and end-to-end loss ratio), they can share a single link measurement implementation. The interface to this module is shown in Figure 6-4.

Chapter 7

ETX Evaluation

This chapter presents experimental results that show that ETX often finds higher-throughput paths than minimum hop-count, particularly between distant nodes. It also explores the effects of a few individual design decisions in the ETX algorithm, and explains why there is a performance gap between the throughput of the routes with the lowest ETX, and the 'best' routes found by searching the network.

We evaluated ETX by running three kinds of experiments. *Routing protocol tests* evaluate how well ETX improved the performance of the DSR and DSDV protocols. *Static throughput tests* show how the underlying throughput of a particular route can change quickly over time, which is a fundamental limitation on how well ETX can predict which route to use. *Single link tests* characterize the accuracy of ETX predictions over a single link at a time, as well as illustrate how delivery ratios of a single link can vary quickly, which also affect how accurately ETX can choose good routes. The following sections describe each set of experiments in more detail.

7.1 Routing Protocol Tests

The routing protocol tests show well ETX improves the throughput of a complete routing system. As a result, they include protocol-specific behavior and overheads. We tested ETX with both the DSDV and DSR routing protocols, which are described further in Chapter 6.

7.1.1 Experimental Setup

Unless otherwise stated, the experimental setup is as follows. The test-bed, radio configuration, and packet size are as described in Section 3.1: 134-byte UDP payloads, 1 mW transmit power, RTS/CTS disabled. The DSDV implementation includes the improvements described in Section 6.2 for both ETX and the hop-count metric. The DSR implementation is as described in Section 6.3.

The protocol performance data presented below were collected during a few separate 'runs'. An entire run takes anywhere from 18 to 72 hours, depending on the experiment parameters. A run considers each pair of nodes in turn. For each pair, one experiment is performed for each routing protocol variant. At the start of each experiment, the routing software is reset (all tables and protocol state are cleared), then the routing protocol and ETX probe algorithm, if ETX is used, are allowed to run long enough to stabilize and setup forwarding and routing state (typically 90 seconds for DSDV). Next, the sending node of the pair sends UDP data packets as fast as the radio allows through the routing system to the destination. The destination counts how many packets arrive over 30 seconds to calculate the average throughput.

After the protocol tests run for each pair, the 'best' static route is identified for that pair by testing the throughput of 10 candidate routes, as described in Section 3.2. Like the routing protocol tests for each pair, the static routes are also tested by sending UDP packets as fast as possible and counting how many are received for 30 seconds. However, packets are forwarded along a static route according to source routes in each packet header, rather than running a dynamic routing protocol. The per-pair protocol interleaving ensures that the results from different routing protocols and the static routing are comparable for the same pair of nodes, since the experiments for each protocol are run within a few minutes of each other for a given pair.

Each graph below is labeled with the run from which it came. Graphs with the same run number are comparable. Graphs with different run numbers should not be compared, since the network's behavior changes substantially with time. The graphs below do not include error bars, but are representative of the many runs performed.

In DSDV experiments using ETX or minimum hop-count, the routing protocol runs for 90 seconds, immediately after which the source sends data packets as fast as possible for 30 seconds. As described in Chapter 5, the heavy load causes the MAC protocol to become extremely unfair, distorting the ETX measurements. To minimize the effects of MAC unfairness, every node routes packets using a snap-

shot of its route table taken at the end of the 90-second warm-up period, before any data is sent. The snapshot also makes the DSDV results more comparable to the 'best' static route results, since the static route tests are not allowed to switch routes in the middle of testing a particular route.

In DSR experiments with ETX or minimum hop-count, a source starts by sending one data packet per second for five seconds. This ensure that DSR sends route requests and finds a route before throughput measurements are taken. After the five seconds pass, the source sends packets as fast as possible for 30 seconds. In DSR experiments with ETX, the source waits an additional 15 seconds before initiating the route request, to give the nodes time to accumulate link measurements.

All experiments run with the appropriate routing overhead. That is, while measuring the throughput of routing with the ETX metric, nodes send periodic ETX broadcast probes. While measuring the throughput of DSDV (with either metric), nodes sends DSDV routing advertisements, just as a production routing system would.

7.1.2 DSDV Performance

Figure 7-1 compares the throughput CDFs of paths found by DSDV using ETX and minimum hop-count, between 100 randomly chosen node pairs. This data is taken from the same run as in Figure 3-2, and shows that DSDV using the ETX metric often finds much faster routes than the minimum hop-count metric.

There are two main regions in Figure 7-1. The right half shows node pairs that could communicate directly, with loss ratios less than about 50% (i.e. with throughput greater than the maximum possible two-hop throughput of 225 packets per second). In these cases the minimum hop-count metric finds the one-hop route, which is the best route, and there is no opportunity for ETX to perform better. The left half corresponds to node pairs with a high direct loss ratio, for which the best route has more than one hop. In this region, the sensitivity of ETX to differences among the many different paths of the same length allows it often to find better paths than hop-count.

Figure 7-2 shows the same data as Figure 7-1, but organized as a scatter plot to allow a direct comparison between the performance of each metric for individual pairs. Each pair is represented by one point; the point's y value is the throughput obtained by DSDV using ETX, and the x value is the throughput obtained by DSDV using minimum hop-count. The upper-right quadrant shows pairs where ETX and minimum hop-count both used the one-hop path.

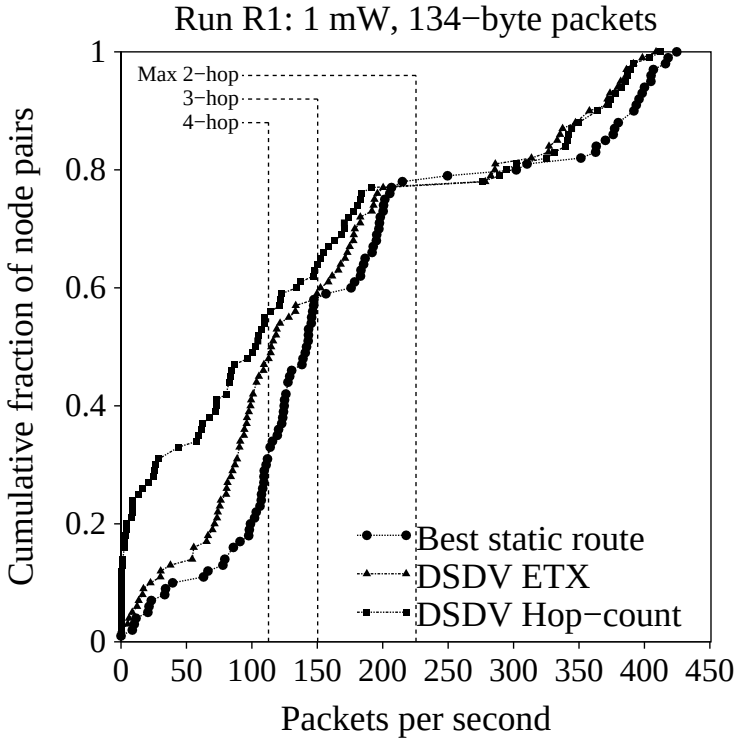


Figure 7-1: ETX finds higher throughput routes than minimum hop-count. This data is taken from the same experimental run as Figure 3-2. Each point represents one of 100 node pairs.

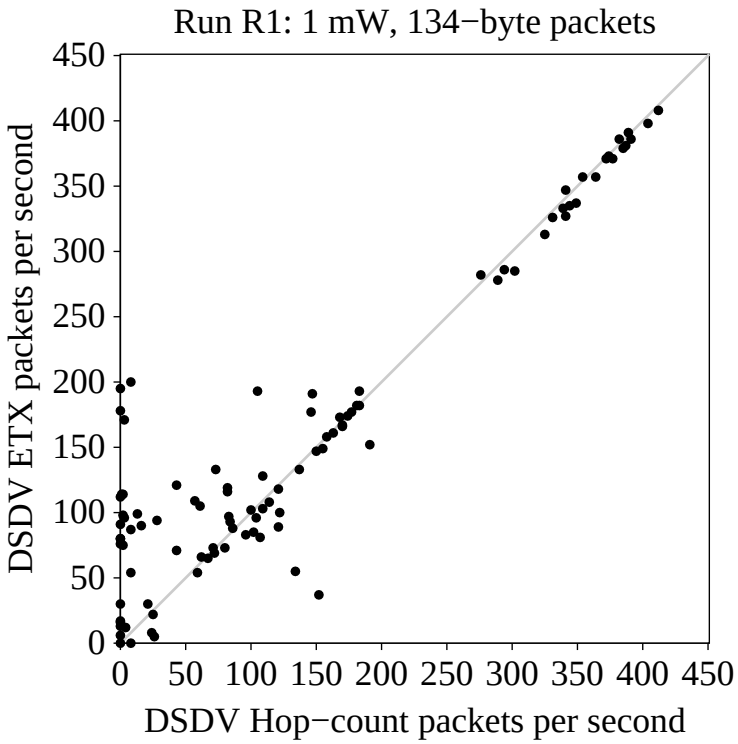


Figure 7-2: The ETX and hop-count data from Figure 7-1, plotted on a per-pair basis. The *x* value of each point shows that pair's throughput for DSDV with minimum hop-count; the *y* value shows the throughput for DSDV with ETX. Points above the line $y = x$ are pairs where ETX outperformed hop-count.

ETX outperforms minimum hop-count by the greatest margin when the hop-count metric uses links with very asymmetric loss ratios. This is illustrated by the points with x near zero and with y relatively large. In these cases, minimum hop-count chooses links that deliver routing updates in one direction but deliver few or no data packets in the other, while ETX correctly avoids those links.

The points for two pairs in Figure 7-2 lie well below the $y = x$ line; this is because of variations in link quality between the ETX and minimum hop-count tests for those pairs. For the first pair, both ETX and hop-count used the same route, so the difference is due to an underlying change in the route's throughput. For the second pair, ETX used a slower 3-hop path while hop-count used a two-hop path; ETX avoided using one of the links in the two-hop path because the measured delivery ratios were very poor. It is likely that the link's quality was different for the ETX and hop-count tests.

ETX incurs more overhead than minimum hop-count, due to its loss-ratio probes, but this overhead is small compared to the gains in throughput that ETX provides. ETX found usable routes for many pairs where minimum hop-count was delivering essentially zero packets per second.

Figure 7-3 shows the throughput CDF for TCP traffic routed using DSDV with ETX and minimum hop-count. The figure also shows the 'best' static route TCP throughput found for each pair. TCP sent data for 30 seconds between each pair. All experimental parameters were the same as for the UDP tests, except that the packet size was varied by TCP according to its congestion control algorithm. Hop-count does particularly poorly for TCP. Although we have not examined the results in as much detail as the UDP results, we conjecture that the hop-count performance suffers for two main reasons. First, since TCP traffic requires good routes in both directions in order to send back end-to-end TCP acknowledgments, there are twice as many chances for hop-count to select a bad route: once in each direction. Second, the TCP back-off algorithm amplifies the effects of any errors in the underlying route: a few lost packets that would not have affected the bulk UDP throughput will cause TCP to greatly decrease its sending rate, resulting in lower throughput.

Figure 7-4 shows the UDP throughput for packets with a 1,386-byte payload. Although ETX still offers an improvement over minimum hop-count, the gain is not as large as for small packets. This is because ETX is still using small probes to estimate the link metrics. Since small packets are more likely to be delivered, ETX is incorrectly over-estimating the quality of each link and causing DSDV to pick sub-optimal routes. For example, if the single-hop direct route between two nodes has an ETX probe delivery rate of 51%, ETX will use it; however, the delivery rate

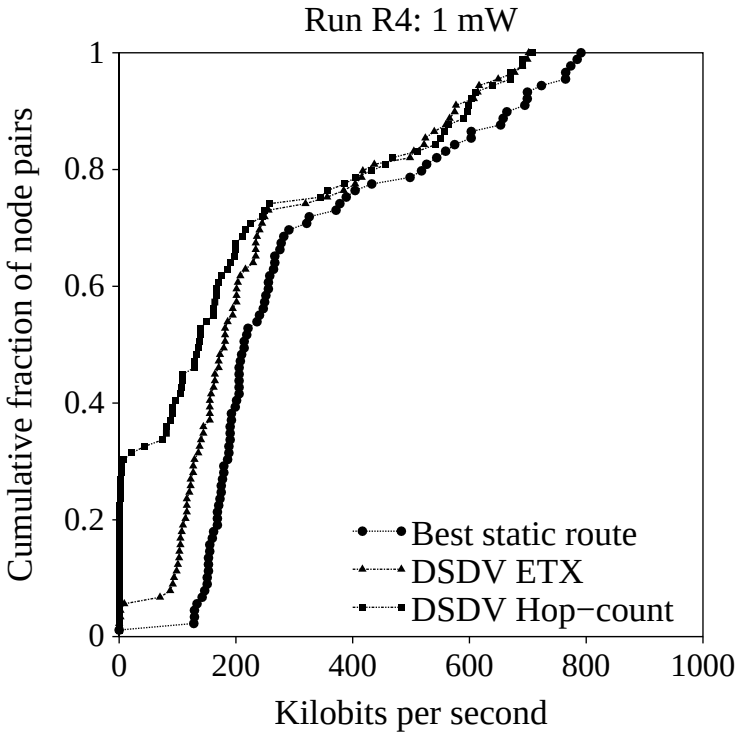


Figure 7-3: ETX finds higher throughput routes than minimum hop-count for TCP traffic. This data is from an experiment similar to Figure 7-1, except that data was sent using the TCP protocol rather than fixed-size UDP packets. The same 100 node pairs are tested as in Figure 7-1.

of 1,386-byte packets on such a link is likely to be much smaller, so a route with a larger number of higher-quality links would have been preferable. The small packets are still useful for detecting very asymmetric links, which is why ETX's gain over minimum is more pronounced to the left of the graph, where hop-count used very asymmetric links.

Figure 7-5 shows the results of ETX versus minimum hop-count from a third run with the radios transmitting at 30 mW instead of 1 mW. The packet size is 134 bytes. When nodes send at the higher transmit power they have more links, as shown in Figure 3-5. This makes the network much more connected, decreasing the average hop-count required for nodes to communicate. As a result, ETX has fewer routes to choose from, and minimum hop-count has a lower chance of choosing a bad route. Figure 7-5 shows that ETX still provides some advantage in the more highly connected network.

Impact of Asymmetry

Some fraction of ETX's gains comes from avoiding extremely asymmetric links. The problem of routing when there are asymmetric links has been addressed in previous work by Lundgren et al. [50] and by Chin et al. [15]. These authors propose a link handshaking scheme to detect and avoid asymmetric links. In this scheme, a node X only accepts route updates from a neighboring node Y if Y is advertising a direct route to X. A node bootstraps the handshake by advertising provisional route entries, which indicate that the node has 'seen' another node, but not yet accepted routes from it.

We implemented the handshaking scheme for DSDV with the minimum hop-count metric. Figure 7-6 compares link handshaking to the ETX and minimum hop-count metrics. Although link handshaking often improves throughput over minimum hop-count alone, ETX finds faster routes. ETX's link measurements allow ETX to discriminate between links with varying degrees of asymmetry and quality.

Effects of DSDV Modifications

Section 6.2 described modifications to DSDV designed to increase its responsiveness to metrics. The *delay-use* modification causes DSDV to delay using a newly received route until it is permitted to advertise the route (i.e. 2×WST has passed). Figure 7-7 shows that the delay-use modification improves the performance of DSDV with ETX.

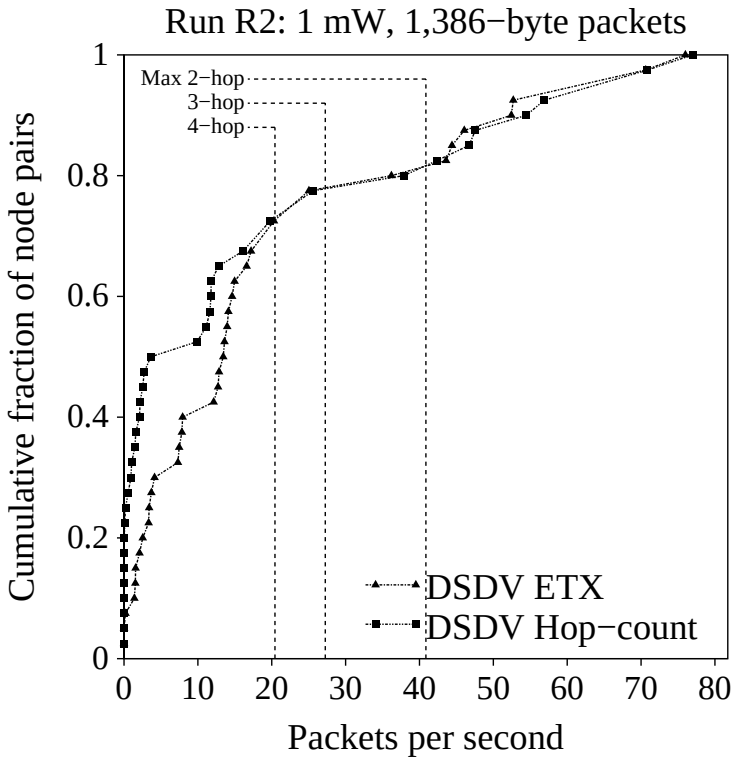


Figure 7-4: ETX provides less of a throughput advantage over minimum hop-count when using large (1,386-byte) packets. The small packets used to measure link loss ratios incorrectly predict the actual transmission counts for large packets. This graph shows 40 pairs randomly chosen from the 100 pairs used in the previous figures. The maximum 1-hop throughput of 1,386-byte data packets at 1 Mbps is 82 packets per second.

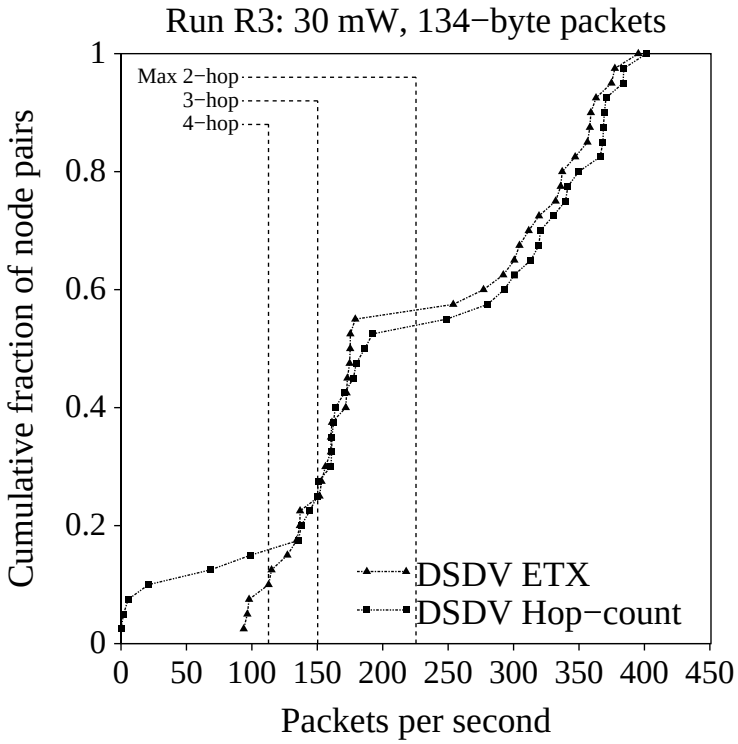


Figure 7-5: ETX versus minimum hop-count when transmitting at 30 mW, for 40 pairs. Using a higher transmit power produces a more highly connected network with many more links and a lower average hop-count, but ETX still provides some advantage.

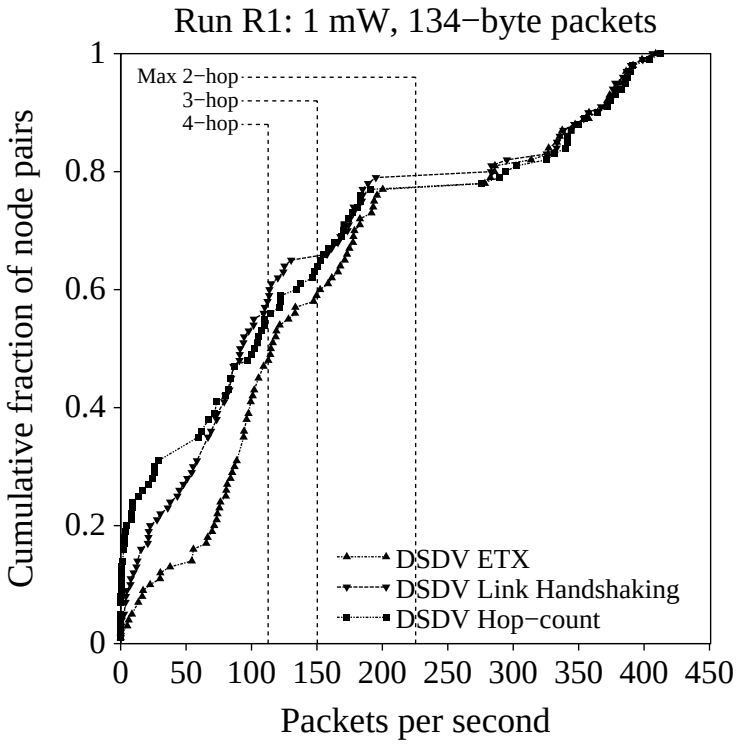


Figure 7-6: ETX provides a significant throughput advantage over a simple handshaking scheme which avoids very asymmetric routes. This is because ETX can make fine-grained decisions between links with varying degrees of asymmetry and quality.

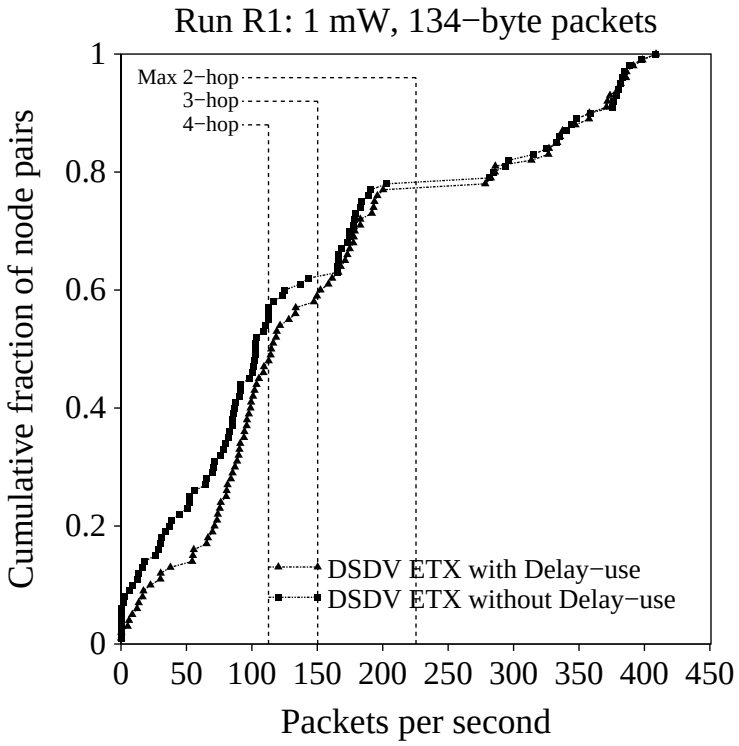


Figure 7-7: DSDV ETX with and without the delay-use modification to DSDV. This modification helps DSDV obey the link metric.

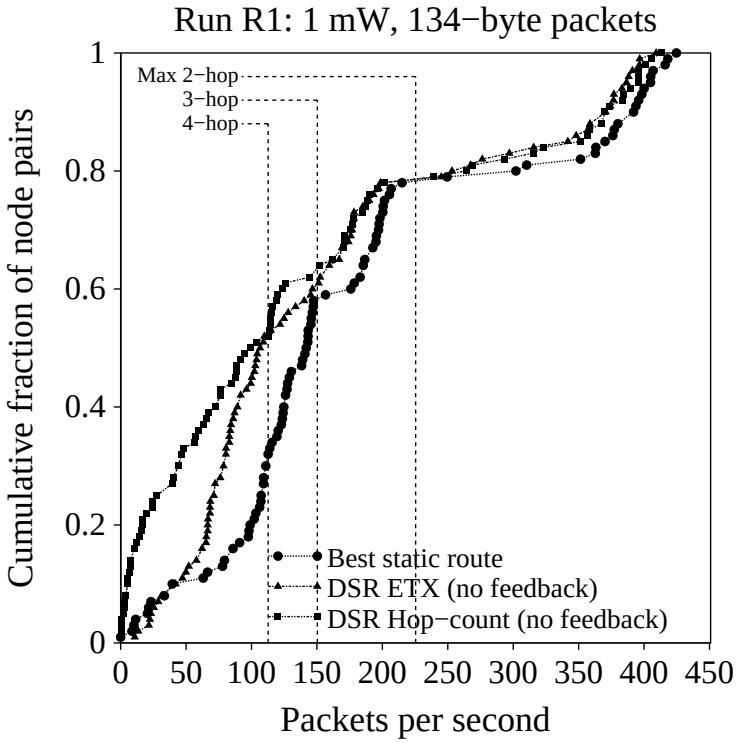


Figure 7-8: Throughput CDFs for DSR ETX compared with DSR hop-count, with link-layer transmission feedback disabled. ETX significantly improves initial route selection.

7.1.3 DSR Performance

This section evaluates the performance of the DSR routing protocol with the ETX metric. As described in Section 6.3, DSR uses link-layer transmission failure feedback to avoid bad routes. To isolate the effects of using ETX with DSR, we evaluated DSR performance both with and without link-layer feedback enabled.

Figure 7-8 shows the effect of using the ETX metric with DSR without link-layer feedback, for the same 100 pairs as in Figure 7-1. Because DSR never learns about transmission failures, no forwarding node ever issues any route errors. Thus DSR uses only the best route found by the initial route request, as determined by the metric.

The figure shows that ETX greatly improves initial route selection in DSR compared to minimum hop-count. This is consistent with the DSDV results in Sec-

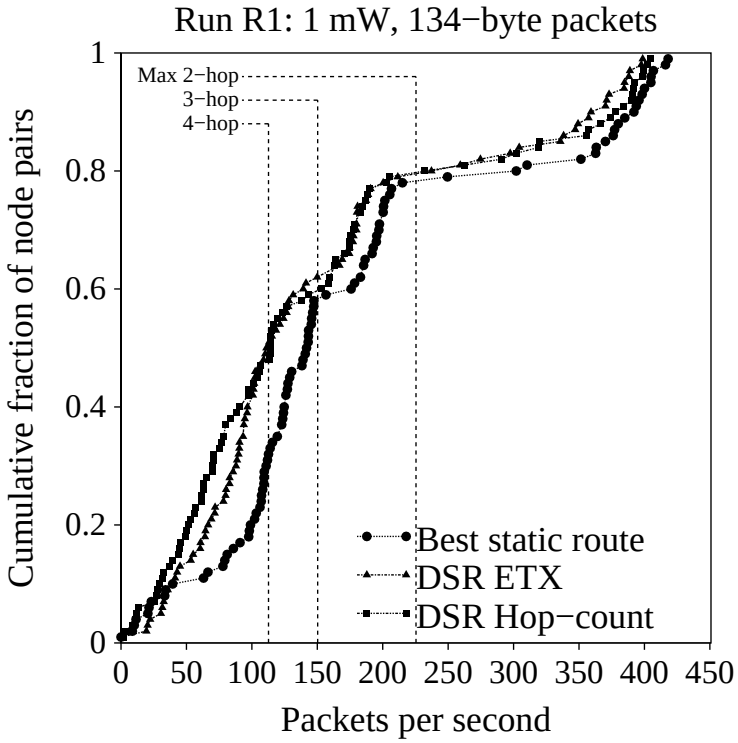


Figure 7-9: DSR ETX compared with DSR hop-count, with link-layer transmission feedback enabled. ETX only slightly improves overall DSR performance, because link-layer transmission failure feedback already helps DSR avoid links with high loss ratios.

tion 7.1.2. Minimum hop-count essentially chooses randomly from all the shortest routes the source obtains from the initial route request; as illustrated in Figure 3.3, this is often not the best route. ETX helps the source pick an initial route with high throughput.

Figure 7-9 illustrates the performance of ETX with DSR's link-layer feedback enabled. ETX provides a small benefit to some pairs in the intermediate and low throughput ranges (the middle and bottom of the CDF). However, failure feedback alone allows DSR to perform almost as well as DSR with ETX.

7.1.4 ETX versus 'Best'

One main question is why there is a gap between the throughput distribution of the routes found using ETX, and 'the' best static routes found by searching the network. Although a route's underlying transmission count does a good job of explaining the route's throughput, routing protocols only use ETX estimates of those transmission counts. ETX mispredicts actual transmission counts due to packet size effects and time variation between when link measurements are made, and when route throughput is tested. Furthermore, the underlying throughput of many routes can change significantly over very short periods of time.

Figure 7-10 shows how the ETX estimates available to the routing protocol mispredict the actual transmission counts for each route. The ETX estimates are calculated using the broadcast delivery ratios for each link measured with the broadcast probes. The actual transmission counts for each route are measured using retransmission counters supplied by the 802.11 radio interface. Because the ETX estimates are often incorrect, the metric can misorder routes, causing the routing protocol to use a route with a lower throughput than the best route available at a given time.

Figure 7-11 shows that if a routing protocol could get a more accurate estimate of the transmission count for each route, the protocol could more accurately choose the highest throughput route. The graph shows the predicted throughput for each route versus the route's actual measured throughput. The predicted throughput is calculated as B/TXC, where B is the maximum link throughput for that packet size (451 packets per second), and TXC is the route's average measured transmission count, obtained from the 802.11 radio interface.

The predicted throughput has a roughly linear relationship to the measured throughput, with some errors, and a systematic offset. The causes of the error and offset are unknown, although there are a few likely suspects about which we can speculate. For example, a route's throughput is affected by both the underlying throughput of its links (determined by their lossiness, and reflected in the transmission count), as well as other 802.11 traffic in the vicinity of those links which contends for the same piece of radio spectrum. That is, a one-hop route over a perfect link might only have half the link throughput if another nearby 802.11 radio was operating. Since our experimental environment is filled with many other 802.11 radios, this case cannot be ruled out. As for the systematic offset, it might be explained by an inaccurate estimate of the maximum ideal throughput for each route. This might happen if the 802.11 interfaces are performing back-off in a slightly different manner than described in the 802.11 specification. Furthermore,

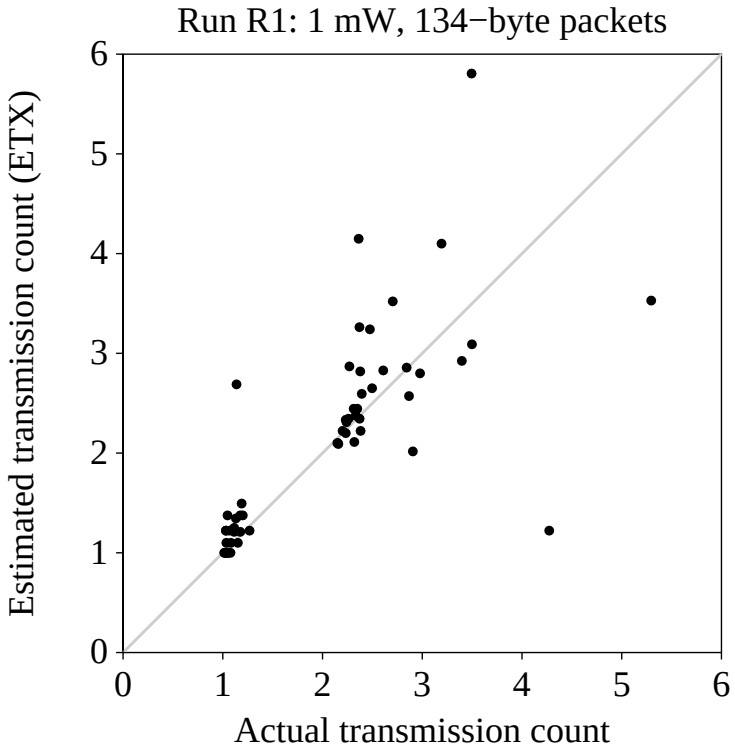


Figure 7-10: The ETX estimates used by DSDV mispredict the actual transmission counts incurred by a route. The *x* value of each point shows the average measured transmission count for a route found by DSDV, while the *y* value shows the ETX metric for that route. This data is from the same run as Figure 7-1.

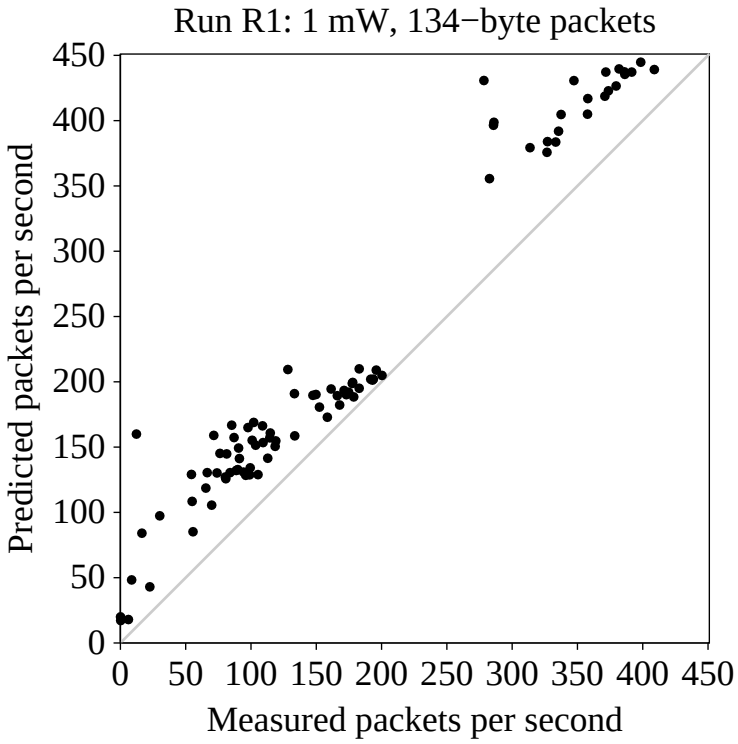


Figure 7-11: A route's underlying transmission counts do a good job of predicting that route's throughput. The *x* value shows the measured throughput of a route found by DSDV using ETX, while the *y* value shows the throughput predicted using the average measured transmission count. This data is from the same run as Figure 7-10.

the throughput predicted using transmission counts does not account for the exponential back-off performed when a radio has to retransmit a packet multiple times.

7.2 Static Throughput Tests

In addition to the routing protocol tests, we ran simple throughput experiments using static routing to explore how the throughput of a particular route changes over time. Each experiment tested the throughput of several routes. For each route, UDP packets were sent as fast as possible for 30 seconds, then after a wait of 30 seconds, packet were again sent as fast as possible for 30 seconds. Figure 7-12 shows the results of these tests for various packet sizes and times of day. During the daytime route throughputs can change substantially, although the throughputs seem very stable at night. One explanation for the daytime variation is the increased level of activity in the lab. People are moving around in lab, opening and closing doors, and running equipment. Another daytime source of variation is the lab's 802.11 access point infrastructure. The 802.11 access points are essentially idle at night, but under heavy use during the day as many lab members and visitors use their laptops to access the lab network wirelessly. This increased and variable 802.11 traffic load can reduce the throughput available to the test-bed.

These experiments show that part of the difference between 'best' and DSDV with ETX in the routing experiments can be explained by underlying variation in the route throughputs. That is, the throughput of the 'best' route may not have been available in the network when the DSDV with ETX experiment was performed. Or, the route selected by DSDV and ETX was no longer the highest throughput route when the data packets were sent. Furthermore, since the 'best' throughput is the maximum throughput of several different routes, is is not sensitive to changes and reordering of those route throughputs: it just selects the highest throughput, and will likely always do better than DSDV. Although the throughput variation does not occur at night, since the route experiments spanned several days, many of them were performed during busy lab hours and would be affected by the route throughput variations.

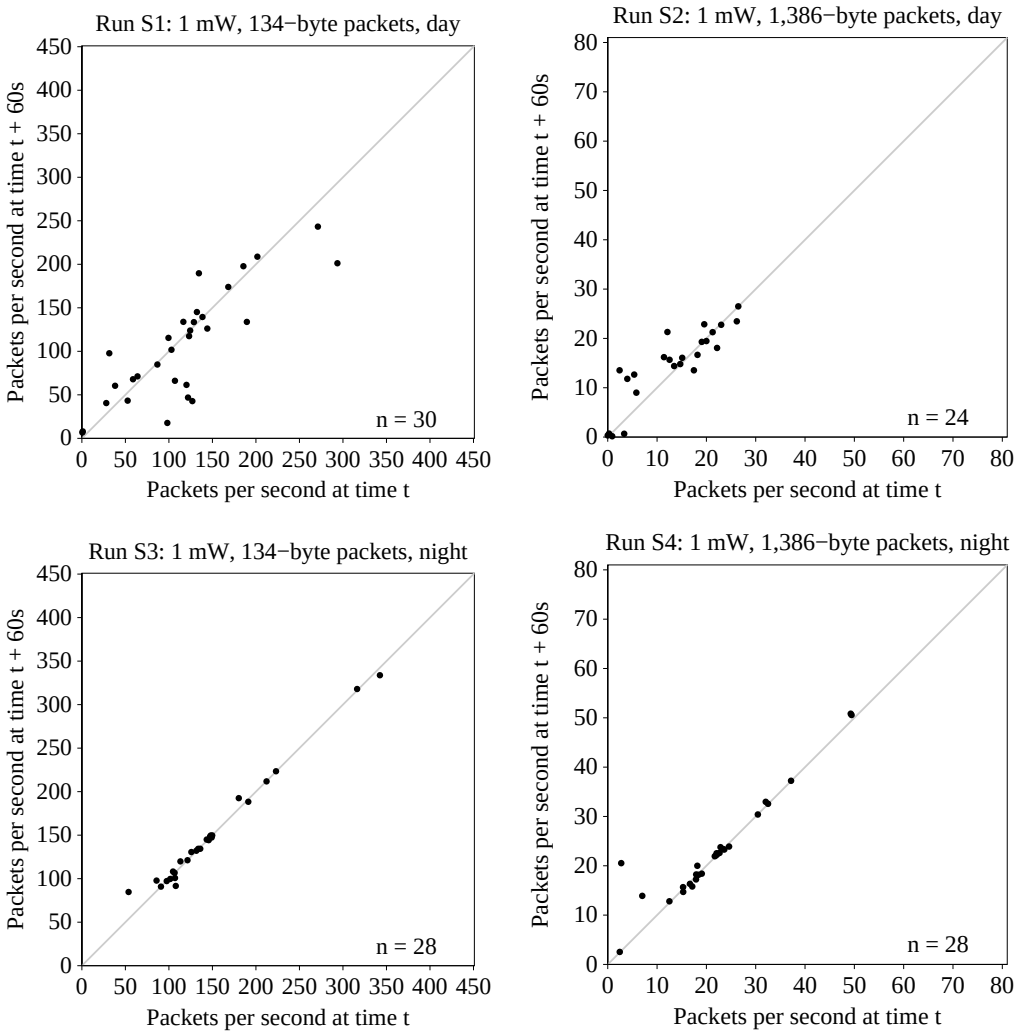


Figure 7-12: The throughput of a route can sometimes vary significantly over a very short period of time. The *x* value for each point shows the throughput of a route measured over 30 seconds. The *y* value shows the throughput of the same route measured 60 seconds later, also over 30 seconds. The left graphs show measurements of 134-byte packets; the right graphs show 1,386-byte packets. The top row shows measurements from the daytime, the bottom row shows measurements from the nighttime.

7.3 Single Link Tests

To better understand the accuracy of link ETX measurements, we performed several experiments with individual links. In each experiment, broadcast delivery ratios are measured to estimate the ETX for a link, then unicast packets are sent to measure the actual transmission count of the link. Broadcasts are typically sent before and after the unicasts to measure how much the delivery ratio of a link varies during the unicast part of each experiment.

Several different links were tested. For each link, the source node sent broadcasts for 20 seconds to the destination node at a fixed rate. The destination node then sent broadcasts to the source node for 20 seconds, also at a fixed rate. Next, the source node sent unicast data packets as fast as possible to the destination node for 30 seconds. Finally, the broadcasts from source to destination and vice-versa were repeated. These experiments did not use any routing protocols, and there was no packet forwarding: packets were only sent over a single link at a time. The broadcast and unicast packets sent by the source node were the same size for each experiment. In a given experiment, the broadcasts sent by the destination were either minimum-size Ethernet packets[1], or the same size as the packets sent by the source.

Figure 7-13 shows how well the ETX estimates actually predict the link transmission counts. The ETX estimates are calculated as described in Equation 5.1, where the values d_f and d_r are measured using the forward and reverse link broadcast tests respectively. The graphs illustrate two main points. First, measuring the reverse ACK delivery ratio of each link (from destination to source) using the minimum-size Ethernet packets provides more accurate ETX estimates, which can be seen by comparing the graphs on the left (data-size reverse measurements) with the graphs on the right (minimum-size reverse measurements). Second, as with the static route throughput tests, there is considerable day-night variation. At night, almost all links work very well, while during the day there is a wider distribution of link performance, along with more short-term time variation of link performance.

Figure 7-14 shows how well ETX estimates actual link transmission counts for larger 1,386-byte packets. The general behavior of the graphs is the same as in

[1] A minimum size Ethernet packet has 14 bytes of Ethernet header, plus 802.11 encapsulation data and headers. The UDP data packets sent as unicasts or broadcast probes also have Ethernet and 802.11 overhead, but that is not included in the packet size. So a minimum-size Ethernet packet is equivalent to a 0-byte UDP packet according to the packet size convention used in this work.

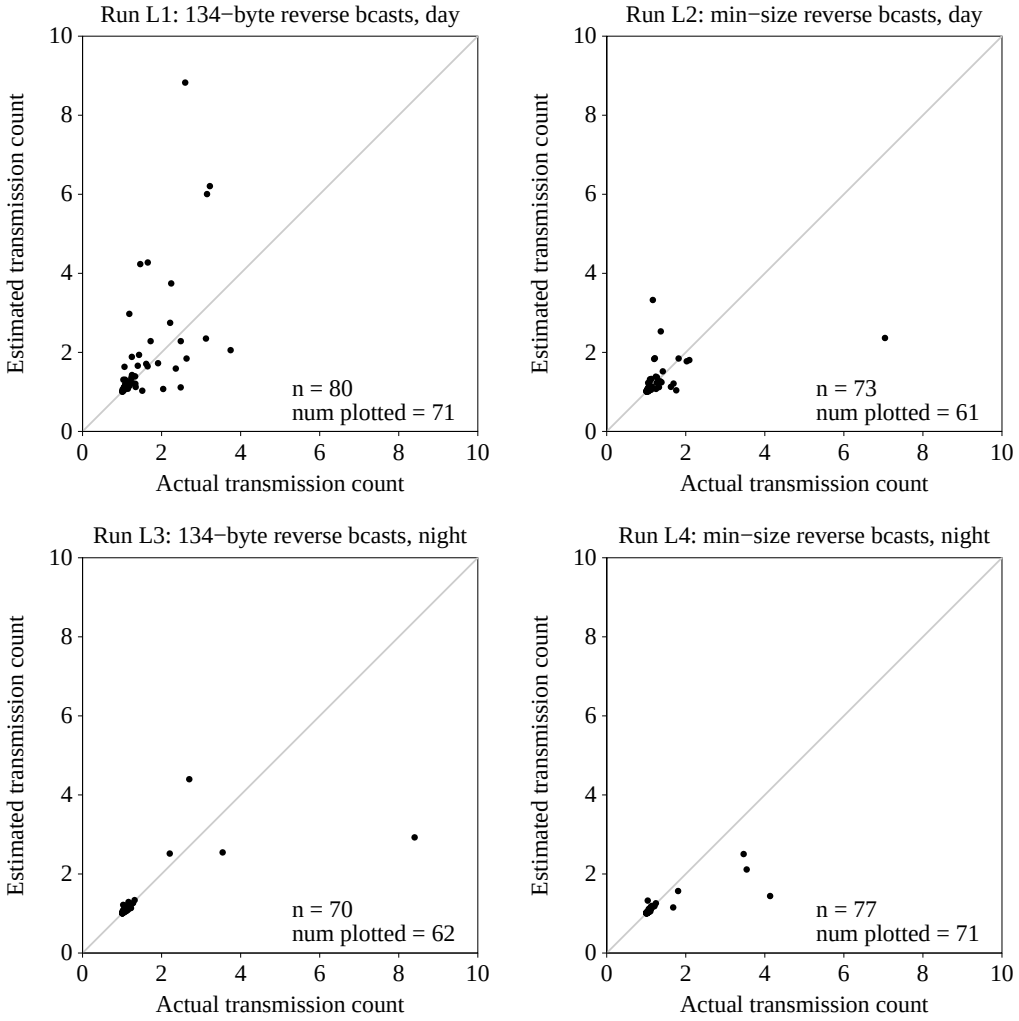


Figure 7-13: ETX versus measured transmission count for 134-byte data packets over a single link. ETX is calculated using the link's broadcast delivery ratio in the forward and reverse directions, measured immediately before sending the unicast data. The left graphs measure the reverse delivery ratio using broadcasts that are the same size as data packets; the graphs on the right use minimum-size Ethernet packets in the reverse direction (14 bytes). The top row shows measurements from the daytime, the bottom row shows nighttime. The nighttime graphs have many points overlapping at (1, 1). Some points are not plotted because they are out of range of the graph.

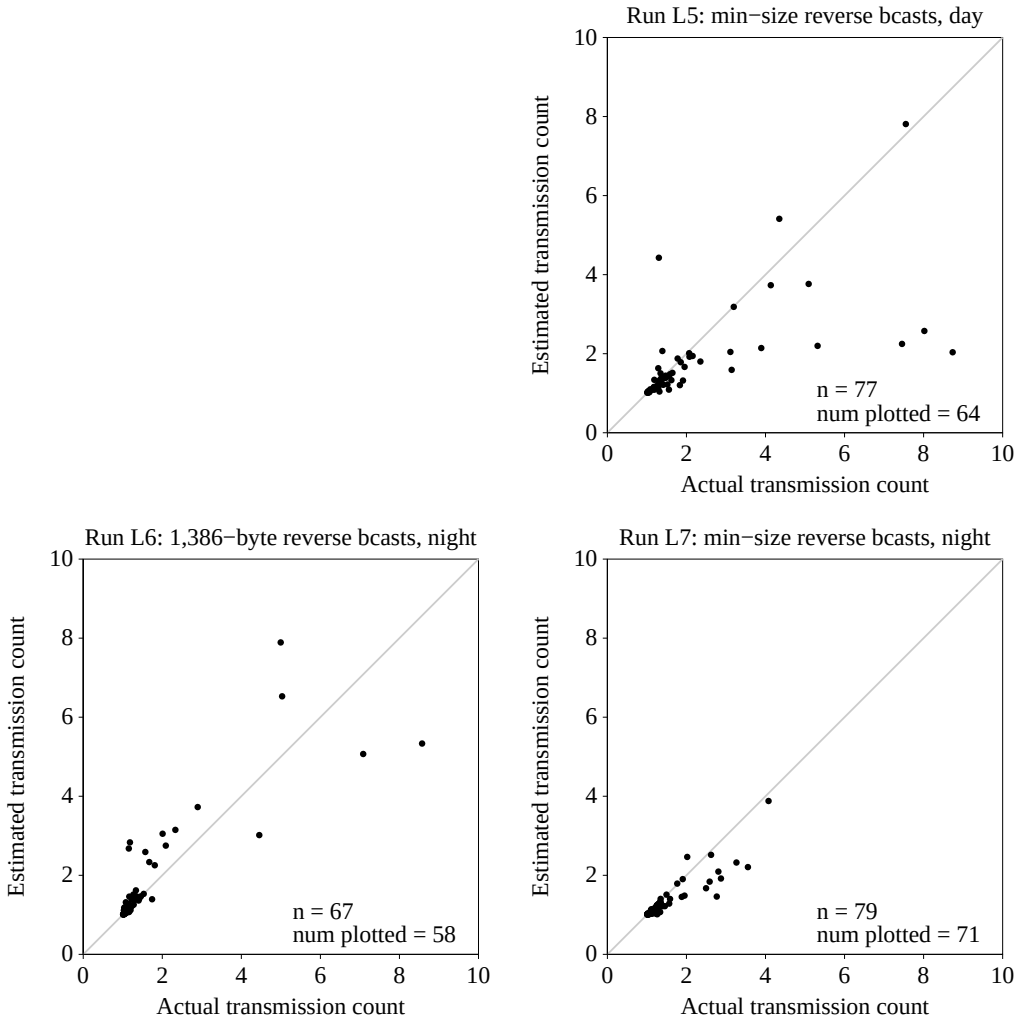


Figure 7-14: ETX versus measured transmission count, as in Figure 7-13, but for 1,386-byte data packets. There is no experiment for measuring the reverse delivery ratio with data-size packets during the daytime. Some points are not plotted because they are out of range of the graph.

Figure 7-13, except that there are still considerable prediction errors even at night. This is likely because the much larger 1,386-byte packets do an even worse job of predicting the ACK delivery ratios in the reverse direction than the 134-byte packets.

Figure 7-15 shows how much link delivery ratios can change over time. The graphs show the measured forward delivery ratio of each link at the beginning and end of each single link test. The graphs show results for both 134- and 1,386-byte packets, during day and night. In all cases there is significant variation between the two measurements. This is less so for the 134-byte packets, which have relatively high delivery ratios across the board. Both the 134- and 1,386-byte sets of experiments show diurnal variation, as in the static throughput tests. The implication of these graphs is that the transmission counts of the links which ETX is trying to estimate can change by a significant amount in a short time. Even if ETX chooses a maximum-throughput route, that route may not be the fastest route 30 seconds later.

7.4 Evaluation Summary

This chapter showed how ETX increases the throughput performance of the DSR and DSDV routing protocols. It also used more focused static throughput and single link experiments to understand the gaps between the throughput of routes found using ETX and the ‘best’ routes found using static routes. We identified two main causes of the discrepancy. First, ETX mispredicts the transmission count of links because it measures the reverse ACK delivery ratios using the wrong packet size. Second, underlying time variations in link delivery ratios and throughputs make it hard for ETX to make accurate predictions.

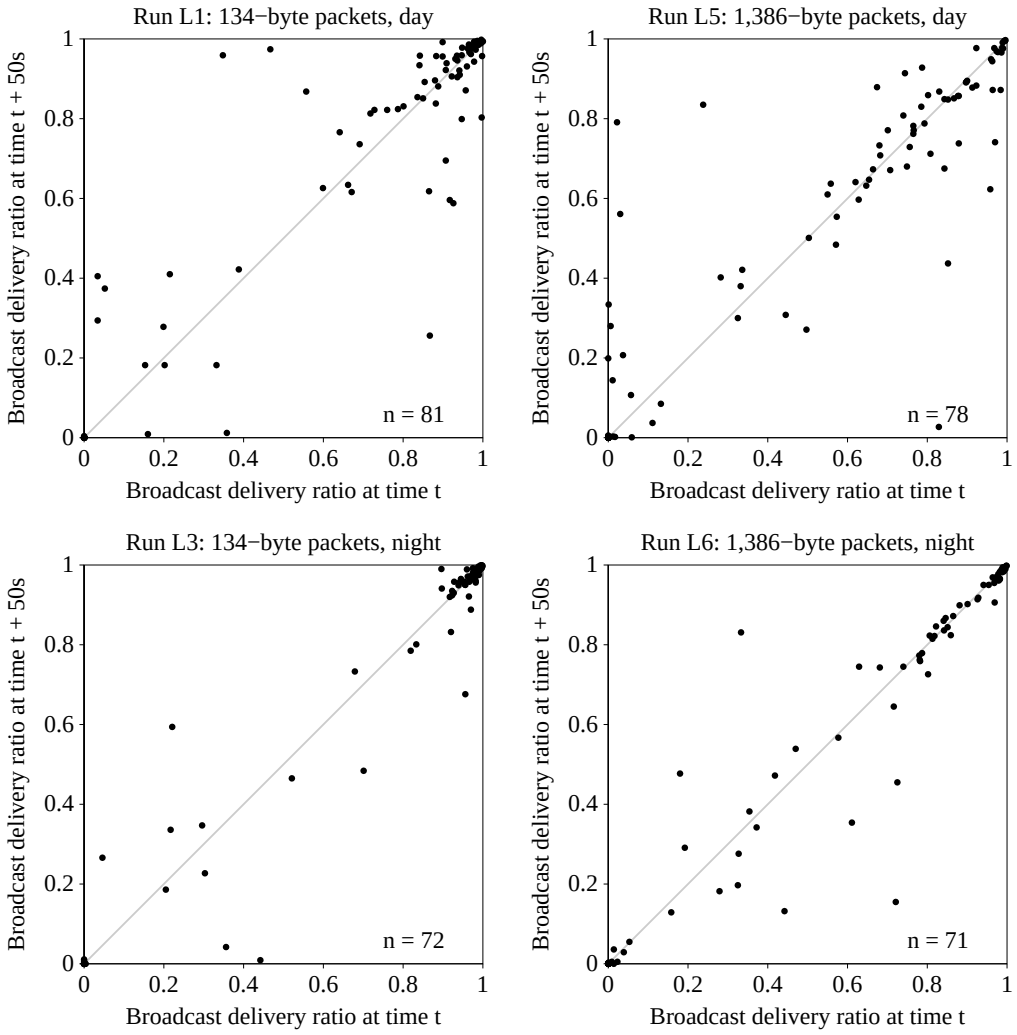


Figure 7-15: The broadcast delivery ratio of a link can change significantly over time. For each link, the graph shows the link's broadcast delivery ratio at some time versus its broadcast delivery ratio 50 seconds earlier. The left graphs show measurements of 134-byte packets; the right graphs show 1,386-byte packets. The top row shows measurements from the daytime, the bottom row shows measurements from the nighttime.

Chapter 8

Future Directions

This chapter outlines possible future work that could expand upon and extend the results in previous chapters. This chapter also describes how ETX fits into the larger design space of multi-hop wireless networks.

8.1 ETX Improvements

This section describes how to improve the performance of ETX and increase its applicability. Many ETX design constraints were chosen to simplify the ETX implementation and design, but can be relaxed with a corresponding increase in complexity.

ETX assumes that all packets are the same size, but the ETX predictions can be adjusted for packet size. One approach would be to use the packet size model in Chapter 4 to correct for ACK and data packet sizes using measurements at only two sizes. Another approach would be to directly measure delivery ratios at all relevant packet sizes, estimating the ACK delivery ratio as the delivery ratio of a minimum-size data packet. In practice this would only involve measuring at two or three packet sizes, since the distribution of data packet sizes has only a few distinct peaks [17]. A related problem is how to modify the routing protocol to handle multiple packet sizes. Because the ETX of each link can vary differently with packet size, each packet size should have its own minimum-ETX route. This can be done simply by including the intended data packet size in each route entry or route control packet (e.g. DSR route requests and replies), at the expense of multiplying the routing state by the number of data packet sizes.

ETX would also benefit from taking into account multiple bit-rates. ETX should be able to trade off a lossy link with high bit-rate for a low-loss, low bit-rate link. This can be done by combining the Medium Time Metric [6] with ETX to produce the *estimated transmission time* (ETT) metric [3]. ETX calculations are simplified by using transmission counts as a proxy for the time each data packet keeps the radio medium busy. ETT makes time explicit, which allows it to combine the ETTs from links with different bit-rates into a single route ETT. ETT can also consider links with different amounts of per-packet overhead.

Many radios can change their bit-rates, including 802.11 radios. This allows links to be made more reliable by decreasing the bit-rate. Nodes can locally decide on the optimal bit-rate for each link by measuring the link at every bit-rate, and choosing the bit-rate which results in the lowest ETT for that link [3]. Another approach would be to treat each physical link as multiple virtual links, one for each bit-rate. The ETT for each virtual link is determined by measuring the link at the specified bit-rate, and the routing protocol finds the best route using the virtual links.

One flaw in the current ETX implementation is that it is highly sensitive to load because of the effects of unfairness and interference on the probe broadcasts. Although a load-sensitive metric might be useful in some applications, ETX is intended to reflect the underlying quality of a route, independent of network traffic. Using a MAC protocol that supports priority traffic might isolate the ETX probe measurements from heavy data traffic, since the data traffic wouldn't prevent ETX probes from being transmitted. Another approach to maintaining accurate transmission count measurements in a busy network would be to use per-packet transmission feedback from the 802.11 interface, which provides a direct measurement of the number of times each data packet is transmitted over each link. Tracking these per-packet data transmissions would incur no extra overhead for routes that are already carrying data traffic.

Because the current implementation of ETX is load sensitive, routing protocols using ETX may oscillate between routes. This can be reduced with appropriate damping [9]. Another approach might be to use multipath routing, choosing paths with the best ETX metrics. Each path will be less loaded than in the single-path case, and will have similar load-sensitive ETX effects, reducing the metric discrepancies which cause the routing protocol to switch routes.

A final problem is how routing protocols should handle route metrics that change over time. The DSDV protocol continuously searches for new routes with better metrics, but has no sense of hysteresis or uncertainty: slight changes in the ETX metric can cause DSDV to switch routes, which can negatively impact

TCP traffic due to reordering. On the other hand, DSR doesn't switch routes unless there are too many retransmissions for a single packet, allowing DSR to keep using a poor route for too long. Routing protocols should notice and react appropriately to changing metrics, without excessive flapping and overhead.

8.2 Wireless Routing and ETX

This section describes larger problems in the area of wireless routing which are related to ETX.

This work did not address mobility, which is important for many wireless networks. One unanswered question is whether or not ETX is effective in a mobile or even partly mobile network. In these networks it is a challenge to distribute accurate topology information and find routes; adding the problem of distributing accurate and consistent route quality metrics like ETX is a further challenge. ETX is likely to be helpful in mobile networks if the routing protocol can obtain accurate metrics and propagate them in a timely manner.

Another problem is how routing metrics and protocols should model the time variation of links and routes. This is a fundamental problem: route metrics must predict the future performance of a route based on past measurements. The ETX design assumes that the loss probability of each link is constant, but this is not realistic. As Chapter 7 showed, the underlying performance of routes and links can change between when a metric is calculated and when a route is used, causing protocols to choose the wrong routes. Protocols might be able to use models of how links change over time to make better predictions. These models could help reduce ETX overhead, for example, by identifying links that don't change much over time, and which require fewer probes to characterize. Also, modeling the uncertainty bounds or time variation of each metric may be useful for choosing routes when the variation of a route's performance is also important [45], as with streaming multimedia traffic, which prefers routes with bounded delay.

This work looked at how ETX improved the performance of sending a single flow at a time. However, most networks have multiple flows of traffic. It may be the case that the minimum-ETX route will not provide the highest throughput to a flow when there is cross-traffic in the network. However, the individual link ETX metrics may be useful as input to higher-level traffic engineering algorithms, such as the work by Jain et al. [36].

An important assumption made by ETX is that each network node has a single radio and antenna, and that the radio uses link-level retries, as in the 802.11

specification. There is a great opportunity to improve network performance by relaxing these assumptions. For example, equipping each node with multiple radios on different bands and allowing them to transmit and receive simultaneously would significantly reduce or eliminate inter-hop interference, if neighboring links are assigned to non-interfering channels [2]. With enough independent channels, the throughput of a route can be made independent of the number of hops in the route, and throughput is likely determined by the bottleneck bandwidth of each link, rather than its ETX. When there is cross-traffic in a network where each link can operate on multiple independent channels, channel assignments for each link should consider both the ordering of link channels within a route, and the interactions between links from different routes which share one or more nodes.

Even though ETX does not predict throughput in networks with multiple flows or multiple channels and radios, it is still useful for increasing total system capacity in busy networks. The capacity of a wireless network with fixed transmission power is determined by its area: the number of transmissions per time per unit area is constant. By choosing minimum-ETX routes for each flow, routing protocols are maximizing the number of packets each flow can send per second. It is not clear, however, what are the fairness properties of ETX in this situation. Also, in networks with many idle regions, total network may be improved by routing some flows out of the way through these idle regions, rather than having them share the throughput of a busy region.

Some wireless networks use end-to-end retransmissions instead of link-level retransmissions. The ETX of a route can be calculated for these networks, albeit in a much more complicated form than Equation 5.3. The exact form of the equation depends on the ordering of the links within each route. ETX will not predict throughput for these networks, because unlike networks with link-layer retransmissions, significant amounts of time can be spent waiting for end-to-end acknowledgments to timeout. However, by incorporating timeout information, the end-to-end ETX can be converted into an end-to-end ETT metric which does predict throughput. And, like networks with multiple radios, ETX is still useful for increasing the total network throughput.

ETX is also useful in networks with variable transmission power. The transmission power at each node is generally decreased as much as possible to reduce energy consumption and increase network capacity, while still keeping the network connected [31, 71]. For any given allocation of transmit powers to nodes, there will likely to be many lossy links, and ETX will be helpful for identifying good routes. And since most radios will have a fixed maximum transmit power, there will always be some links that have marginal performance.

Using more sophisticated radio technology may reduce some of the underlying link loss problems that routing protocols can detect using ETX. Improvements in radio coding and modulation, and multi-user access can improve packet delivery ratios over a link. For example, the orthogonal frequency-division multiplexing (OFDM) [16] technique adaptively chooses sets of frequencies to avoid noise and interference, thereby decreasing packet error rates. However, this incurs a throughput cost because OFDM is avoiding some fraction of the available radio spectrum. Each link has an intrinsic throughput related to its individual RF characteristics and interference, which cannot be exceeded. Increasingly advanced modulation or coding techniques will only allow radio systems to obtain a larger fraction of the fundamental throughput of each link. So, there will alway be an opportunity to use a metric like ETX or ETT to find high-throughput routes using the best links in the network.

Chapter 9

Related Work

Much of the recent work in ad hoc routing protocols for wireless networks [61, 37, 62] has focused on coping with mobile nodes, rapidly changing topologies, and scalability. In general this work has been carried out using analysis and simulation that focuses on how mobility affects link connectivity and routing protocol behavior. To this end, researchers have used link connectivity models based on radio ranges: in-range links work well, while out-of-range links are broken. This model works well when the dominating factor in determining link connectivity is node location and motion. However, this on-off link model is not as useful for fixed networks, where the vagaries of individual link performance are no longer dwarfed by the effects of motion. In particular, protocols that seem promising for simulated mobile networks often don't provide the best performance for fixed networks, as Chapter 3 illustrated. Many of the ways that simulation and analysis could be altered to model wireless links more accurately are detailed by Kotz et al. [46]. Recent protocol design and evaluation work has started to focus on the detailed behavior of real wireless links.

The behavior of routing protocols over lossy links has been addressed and evaluated by real implementations in several recent papers. Lundgren et al. [50] relate their experiences with a 802.11-based multi-hop network with four nodes, and they coin the term 'gray zones' to refer to links that deliver routing protocol packets but not data packets. They propose using link handshaking and counting route broadcasts to filter out gray zone links. Link handshaking requires both ends of a link to acknowledge the other end before using the link. Chin et al. [15] also describe a four-node multi-hop network based on 802.11 radios, and independently propose link handshaking to filter out asymmetric links.

The CMU/Rice Monarch project has had several years of experience with their DSR implementation [25], which has provided important lessons about wireless network emulation [41], and the performance of real implementations [51]. Hu and Johnson describe an eight-node mobile test-bed using DSR that reliably transmits video and audio streams [34]. Because mobility causes new links to form and older links to break, they modify DSR to preemptively issue DSR route requests when a link's signal-to-noise ratio (SNR) drops below a given threshold. The assumption is that links with low SNRs are likely to break soon because of motion. Preemptively issuing a route request allows the source node to have a fresh route ready to use before the link actually does break. However, even with this modification, DSR does not discriminate between links that are deemed functional: it treats all working links as equivalent, and finds minimum hop-count routes.

The ideas presented in this work are motivated by experiments on a relatively large-scale test-bed, with 5 or 6 times more nodes and an order of magnitude more links than the test-beds used in most previous work. Chapter 3 showed that these links have many different qualities, evenly spread from best to worst, and that there is no easy division of links into 'good' and 'bad' categories. However, experiments on smaller test-beds are not as likely to reveal such a wide distribution of link quality, and previous work has been more focused on categorizing links as 'good' or 'bad'. In contrast, the mechanisms and protocols introduced in this work are specifically designed to accommodate and take advantage of links of all qualities.

Some of the earliest and most important work in multi-hop wireless networking was the DARPA Packet Radio Network (PRNet) research [39], which was carried out in the 1970s and 1980s.[1] The PRNet was specially designed for multi-hop wireless networking at all layers of the system, including a new spread-spectrum packet radio design [28]. The system was also designed to handle lossy links, and used link-level packet loss ratio measurements to quantify link quality. However, the PRNet used loss ratio thresholds to distinguish good links from bad, which has a few drawbacks, as discussed below. PRNets were deployed and experimented with by many research groups, at a scale similar to the test-bed network used in this work [29].

One unanswered question is whether or not the PRNet deployments had similar link loss rate distributions to those shown in Chapter 3. It is conceivable that the different radio design and deployment scenarios produced a different distribu-

[1] Many more papers related to the PRNet can be found in a special issue of the IEEE Proceedings [1].

tion of link loss ratios, and that the threshold technique used by the PRNet was indeed the most suitable technique.

The PRNet research also produced a new routing metric, called Least Interference Routing (LIR) [74]. This was shown in simulation to increase the total network throughput of random packet radio networks. Routing protocols using LIR select routes that interfere with the fewest number of other nodes, by choosing routes to minimize the sum of the number of one-hop neighbors of each node in the route. LIR differs from the ETX metric presented here in that it does not directly evaluate actual link performance, in terms of packet loss ratios; instead, it uses the interference metric to predict performance.

There is also much related work in the field of sensor networks, even though on the face of it sensor networks are very different from the multi-hop wireless networks we are concerned with. In general, the nodes in sensor networks have orders of magnitude fewer resources than a typical PC, in terms of processing power, memory, and available energy. In addition, because of size and power constraints, sensor radios are much less sophisticated than 802.11 radios, and the application requirements are much less: sensor networks typically carry very low-bandwidth data streams. However, recent work has shown that despite these differences, sensor networks must deal with the same sort of link lossiness and variation that multi-hop wireless data networks face. Indeed, the ETX metric presented here is useful in both sorts of networks.

Zhao and Govindan [80] provide a detailed analysis of the performance of wireless links in three different network deployment scenarios, using Berkeley Motes, a low-power sensor network platform. They find that many links can be lossy to varying degrees, and, as in Chapter 5, find that signal strength is not an adequate predictor of packet loss ratios, because of effects such as multipath interference.

Woo et al. [78] also provide measurements showing the variability of links in a Mote-based network. They use the measurements to drive an analysis of link estimators and routing protocol techniques. They propose the *Minimum Transmission* (MT) metric, and show that it greatly improves the end-to-end packet delivery ratio in their experiments compared to minimum hop-count. MT is the same metric as ETX, except that the link delivery ratios are estimated by passively snooping on data packets over a fixed time window, rather than sending special probes. Furthermore, the link measurements are smoothed with an exponentially-weighted moving average, and a hysteresis is applied when deciding whether to switch routes based on a metric change.

Yarvis et al. [79] also observe that hop-count performs poorly as a routing metric for a Mote-based sensor network. They present a path metric which approximates the product of the per-link delivery ratios. As argued in Chapter 5, this metric is likely to use low-loss paths with many hops in situations where a path with a smaller number of higher loss links would perform better.

A number of existing ad hoc wireless routing algorithms collect per-link signal strength information and apply a threshold to avoid links with high loss ratios [15, 22, 26, 30, 34, 39, 50, 67]. One problem with these approaches is that given the complexities of radio links such as multipath interference and fading, it is unlikely that SNR measurements can be used to accurately predict packet loss ratios; this is shown experimentally in Chapter 5 for 802.11 radios and by Zhao and Govindan for Motes [80]. It is true that SNR measurements provide some useful information about the underlying quality of a link: links with extremely poor SNR values are likely to have high packet loss ratios. But, it is not clear what the appropriate SNR threshold should be. Setting the SNR threshold too high may eliminate links that are necessary for connectivity, while setting the threshold too low will allow the routing protocol to use poor links with low throughput. In general, any approach that uses thresholds will suffer a similar problem. For example, Woo et al. [78] show that a routing protocol which simply ignores links with delivery ratios below a given threshold is either unable to keep the network connected, or unable to find routes with good end-to-end delivery ratios, depending on the threshold used. ETX avoids these problems by allowing any link to be directly compared to another link, and by assigning metrics that allow all routes to be compared and ranked. This avoids leaving out any links that are required to connect the network, but still enables the routing protocol to choose better links and routes over worse.

A complementary solution to high link loss ratios is to improve the effective loss ratio with some form of redundancy. Forward error correction, MAC-level acknowledgment and retransmission, and solutions such as Snoop-TCP [7] and Tulip [60] all take this approach. However, even with these techniques it is preferable to use links with low loss ratios rather than links with high loss-ratios: retransmissions (or other redundancy) reduce useful link capacity and increase interference.

Like most of the work discussed above, this work treats wireless networks from the bottom up, trying to find good techniques and abstractions for characterizing and distinguishing between links and routes in wireless networks. However, there is a large body of work that is concerned with Quality of Service (QoS) in networks, especially wireless multimedia networks. Wireless QoS algorithms approach route selection from the top down. Some techniques explicitly schedule

transmission slots in time or frequency division MAC layers to provide bandwidth guarantees [14, 33, 49, 53, 81], while others treat the MAC as opaque, and rely upon it for bandwidth and delay information and constraints [13, 70, 72]. These approaches are only successful if the lower layers can provide accurate information about the actual links, such as the average number of usable transmission slots, or the achievable throughput of each link. Unfortunately, this link information is hard to determine in the sort of distributed multi-hop networks we are concerned with, because of unknown data traffic patterns or network topology, and inter-node interference. Indeed, although the ETX metric enables routing protocols to find high-throughput routes, it doesn't provide enough information about links to QoS algorithms to allow them to make bandwidth or latency guarantees, especially when multiple nodes are sending data.

We assume that the loss ratio of a given link cannot be controlled by the system. More sophisticated hardware might allow transmit power levels to be changed to make links better behaved. Existing systems exploit this idea, often with a focus on minimizing the energy consumption required to successfully deliver data [32, 40, 68]. Energy consumption is primarily a concern for sensor networks, where radio transmissions consume the majority of each node's energy budget. Fixed data networks like those we consider are not likely to be concerned with energy consumption. However, fixed networks can benefit by using power control to reduce transmission ranges and increase network capacity [31, 71].

Since the ETX metric assumes that all links run at the same bit-rate, it does not properly find high-throughput routes when links run at multiple bit-rate. Awerbuch et al. [6] present the *Medium Time Metric* to help find high-throughput paths when links can run at different bit-rates. Since their metric does not account for losses, it is complementary to ETX.

Chapter 10

Conclusion

The main contribution of this work is a simple way for multi-hop wireless routing protocols to choose high-throughput paths in networks with link-layer retransmissions. By measuring the delivery ratios of each link in a route using fixed size broadcast packets, protocols can estimate the throughput of the route as the inverse of that route's expected transmission count, which is called ETX. The ETX of a route R is calculated as

$$\mathrm{ETX}(R) = \sum_{i \in R} \frac{1}{d_f^i \times d_r^i} \tag{10.1}$$

where d_f^i and d_r^i are the measured delivery ratios in the forward and reverse directions of each link i in R.

The inverse of ETX predicts throughput for routes with small to medium hop-counts. Since at most one node in those routes can transmit at any time, the throughput of the route is limited by the number of transmissions, or equivalently, time, required to transmit each packet over the route. Measurements on a real testbed network show that ETX helps the DSR and DSDV routing protocol find routes with significantly higher throughput than the default minimum hop-count metric. The overhead of the ETX delivery ratio probes depends on the spatial density of the network, and is relatively small compared to the amount of data traffic that can be sent over each link.

This work also characterized the delivery ratios and asymmetry of the testbed network, and showed how lossy and asymmetric links affect route throughput. Lossy links require more retransmissions, and therefore have lower effective throughput. However, a route with few lossy links can be preferable to a route

with many higher-quality links, since contention between links also reduces route throughput.

Finally, this work proposed a simple model for how link delivery ratios vary with packet size. The packet delivery P_p for a packet with n data symbols is

$$P_p(n) = P_f \times P_s^n \tag{10.2}$$

where P_f is the per-packet probability that a receiver successfully acquires and synchronizes to a packet frame, and P_s is the per-symbol probability that the receiver successfully decodes that symbol. Measurements on the test-bed show that this model can accurately predict the delivery ratios at many packet sizes using measurements at two packet sizes over each link.

Bibliography

[1] *Proceedings of the IEEE*, 75(1), January 1987. Special Issue on Packet Radio Networks.

[2] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. Technical Report MSR-TR-2003-44, Microsoft Research, July 2003.

[3] Daniel Aguayo, John Bicket, and Robert Morris. SrcRR: A high throughput routing protocol for 802.11 mesh networks. In preparation, 2004.

[4] Daniel Aguayo, Sanjit Biswas, John Bicket, and Robert Morris. A measurement study of a rooftop 802.11b mesh network. In preparation, 2004.

[5] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[6] Baruch Awerbuch, David Holmer, and Herbert Rubens. High throughput route selection in multi-rate ad hoc wireless networks. Technical report, Johns Hopkins University, Computer Science Department, March 2003. Version 2.

[7] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving TCP performance of wireless links. *IEEE/ACM Transactions on Networking*, 6(5), December 1997.

[8] Brahim Bensaou, Yu Wang, and Chi Chung Ko. Fair medium access in 802.11 based wireless ad-hoc networks. In *First Annual IEEE and ACM International Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc)*, August 2000.

[9] Dimitri P. Bertsekas. Dynamic behavior of shortest path routing algorithms for communication networks. *IEEE Transactions on Automatic Control*, AC-27(1):60–74, February 1982.

[10] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACAW: A media access control protocol for wireless LANs. In *Proc. ACM SIGCOMM Conference (SIGCOMM '94)*, pages 212–225, August 1993.

[11] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM/IEEE MobiCom*, pages 85–97, October 1998.

[12] Alberto Cerpa, Naim Busek, and Deborah Estrin. SCALE: A tool for simple connectivity assessment in lossy environments. Technical Report 0021, UCLA Center for Embedded Network Sensing (CENS), September 2003.

[13] Shigang Chen and Klara Nahrstedt. Distributed quality-of-service routing in ad hoc networks. *IEEE Journal on Selected Areas in Communcations*, 17(8), August 1999.

[14] T.-W. Chen, J.T. Tsai, and M. Gerla. QoS routing performance in multihop, multimedia, wireless networks. In *Proceedings of IEEE ICUPC '97*, 1997.

[15] Kwan-Wu Chin, John Judge, Aidan Williams, and Roger Kermode. Implementation experience with MANET routing protocols. *ACM SIGCOMM Computer Communications Review*, 32(5), November 2002.

[16] Leonard J. Cimini, Jr. Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing. *IEEE Transactions on Communications*, COM-33(7), July 1985.

[17] k claffy, Greg Miller, and Kevin Thompson. The nature of the beast: Recent traffic measurements from an Internet backbone. In *Proceedings of INET '98*. The Internet Society, July 1998.

[18] IEEE Computer Society LAN/MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. The Institute of Electrical and Electronic Engineers, New York, New York, 1997. IEEE Std. 802.11–1997.

[19] IEEE Computer Society LAN/MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band*. The Institute of Electrical and Electronic Engineers, New York, New York, 1999. IEEE Std. 802.11a–1999.

[20] IEEE Computer Society LAN/MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*. The Institute of Electrical and Electronic Engineers, New York, New York, 1999. IEEE Std. 802.11b–1999.

[21] IEEE Computer Society LAN/MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band*. The Institute of Electrical and Electronic Engineers, New York, New York, 2003. IEEE Std. 802.11g–2003.

[22] Brian H. Davies and T. R. Davies. The application of packet switching techniques to combat net radio. In *Proceedings of the IEEE* [1]. Special Issue on Packet Radio Networks.

[23] D. M. J. Devasirvatham, M. J. Krain, and D. A. Rappaport. Radio propagation measurements at 850 MHz, 1.7 GHz and 4 GHz inside two dissimilar office buildings. *Electronics Letters*, 26(7):445–447, March 1990.

[24] Sheetalkumar Doshi, Shweta Bhandare, and Timothy X. Brown. An on-demand minimum energy routing protocol for a wireless ad hoc network. *Mobile Computing and Communications Review*, 6(2), July 2002.

[25] CMU/Rice Monarch DSR Source Code Distribution. `http://monarch.cs.rice.edu/dsr-impl.html`.

[26] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi. Signal stability-based adaptive routing (SSA) for ad hoc mobile networks. *IEEE Personal Communications*, February 1997.

[27] Dan Duchamp and Neil F. Reynolds. Measured performance of a wireless LAN. In *Proceedings of the 17th Annual Conference on Local Computer Networks*, September 1992.

[28] William C. Fifer and Frederick J. Bruno. The low-cost packet radio. In *Proceedings of the IEEE* [1]. Special Issue on Packet Radio Networks.

[29] M. S. Frankel. Advanced technology testbeds for distributed, survivable command, control, and communications. In *Proceedings of the IEEE MILCOM Conference*, volume 3, October 1982.

[30] Tom Goff, Nael B. Abu-Ghazaleh, Dhananjay S. Phatak, and Ridvan Kahvecioglu. Preemptive routing in ad hoc networks. In *Proc. ACM/IEEE MobiCom*, July 2001.

[31] Piyush Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions of Information Theory*, 46(2):388–404, March 2000.

[32] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient Communication Protocols for Wireless Microsensor Networks. In *Proceedings of the Hawaiian International Conference on Systems Science*, January 2000.

[33] Yu-Ching Hsu, Tzu-Chieh Tsai, Ying-Dar Lin, and Mario Gerla. Bandwidth routing in multi-hop packet radio environment. In *Proceedings of the 3rd International Mobile Computing Workshop*, 1997.

[34] Yih-Chun Hu and David B. Johnson. Design and demonstration of live audio and video over multihop wireless ad hoc networks. In *Proceedings of the MILCOM*, 2002.

[35] *HFA3863 Direct Sequence Spread Spectrum Baseband Processor with Rake Receiver and Equalizer Data Sheet*. Intersil Americas Inc., December 2001.

[36] Kamal Jain, Jitendra Padhye, Venkat Padmanabhan, and Lili Qiu. Impact of interference on multi-hop wireless network performance. In *Proc. ACM/IEEE MobiCom*, September 2003.

[37] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[38] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The Dynamic Source Routing protocol for mobile ad hoc networks (DSR). Internet draft (work in progress), IETF, April 2003. `http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt`.

[39] John Jubin and Janet D. Tornow. The DARPA packet radio network protocols. In *Proceedings of the IEEE* [1]. Special Issue on Packet Radio Networks.

[40] Eun-Sun Jung and Nitin Vaidya. A power control MAC protocol for ad hoc networks. In *Proc. ACM/IEEE MobiCom*, September 2002.

[41] Qifa Ke, David A. Maltz, and David B. Johnson. Emulation of multi-hop wireless ad hoc networks. In *Proceedings of the Seventh International Workshop on Mobile Multimedia Communications (MOMUC 2000), IEEE Communications Society*, October 2000.

[42] Atul Khanna and John Zinky. The revised ARPANET routing metric. *ACM SIGCOMM Computer Communication Review*, 19(4), September 1989.

[43] Ralf Koetter and Muriel Medard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, October 2003.

[44] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(4), November 2000.

[45] C. Emre Koksal and Hari Balakrishnan. Quality-aware routing in time-varying wireless networks. In preparation, 2004.

[46] David Kotz, Calvin Newport, and Chip Elliott. The mistaken axioms of wireless-network research. Technical Report TR2003-467, Dept. of Computer Science, Dartmouth College, July 2003.

[47] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, pages 61–69, Rome, Italy, July 2001.

[48] Shuo-Yen Robert Li, Raymond W. Yeung, and Ning Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, February 2003.

[49] Chunhung Richard Lin. On-demand QoS routing in multihop mobile networks. In *Proc. IEEE Infocom*, April 2001.

[50] Henrik Lundgren, Erik Nordström, and Christian Tschudin. Coping with comunication gray zones in IEEE 802.11b based ad hoc networks. In *5th ACM international Workshop on Wireless Mobile Multimedia (WoWMoM 2002)*, September 2002.

[51] David A. Maltz, Josh Broch, and David B. Johnson. Quantitative lessons from a full-scale multi-hop wireless ad hoc network testbed. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, September 2000.

[52] James L. Massey. Optimum frame synchronzation. *IEEE Transactions on Communications*, COM-20(2), April 1972.

[53] Anastassios Michail and Anthony Ephremides. Algorithms for routing session traffic in wireless ad-hoc networks with energy and bandwidth limitations. In *Proceedings of 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 2001.

[54] Eytan Modiano. An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols. *Wireless Networks*, 5(4):279–286, July 1999.

[55] Thyagarajan Nandagopal, Tae-Eun Kim, Xia Gao, and Vaduvur Bharghavan. Achieving MAC layer fairness in wireless packet networks. In *Proc. ACM/IEEE MobiCom*, pages 87–98, August 2000.

[56] Giao T. Nguyen, Randy H. Katz, Brian Noble, and Mahadev Satyanarayanan. A trace-based approach for modeling wireless channel behavior. In *Proceedings Winter Simulation Conference '96*, December 1996.

[57] P. Nobles, D. Ashworth, and F. Halsall. Indoor radiowave propagation measurements at frequencies up to 20 GHz. In *Proceedings 44th IEEE Vehicular Technology Conference*, June 1994.

[58] P. Nobles and F. Halsall. Delay spread and received power measurements within a building at 2 GHz, 5 Hz and 17 GHz. In *Proceedings 10th International Conference on Antennas and Propagation*, April 1997.

[59] The Network Simulator — ns-2, 2003. `http://www.isi.edu/nsnam/ns`.

[60] Christina Parsa and J. J. Garcia-Luna-Aceves. TULIP: A link-level protocol for improving TCP over wireless links. In *Proc. IEEE Wireless Communications and Networking Conference 1999 (WCNC 99)*, September 1999.

[61] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proc. ACM SIGCOMM Conference (SIGCOMM '94)*, pages 234–244, August 1993.

[62] Charles E. Perkins and Elizabeth M. Royer. Ad hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[63] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein. Theory of spread spectrum communications—A tutorial. *IEEE Transactions on Communications*, 20(5):855–884, May 1982.

[64] R. Price and P. E. Green. A communication technique for multipath channels. *Proceedings of the IRE*, 46:555–570, March 1958.

[65] John G. Proakis. *Digital Communications*. McGraw-Hill, 2001.

[66] Rice Monarch Project. Wireless and mobility extensions to ns-2. `http://www.monarch.cs.rice.edu/cmu-ns.html`.

[67] Ratish J. Punnoose, Pavel V. Nitkin, Josh Broch, and Daniel D. Stancil. Optimizing wireless network protocols using real-time predictive propagation modeling. In *Radio and Wireless Conference (RAWCON)*, August 1999.

[68] Ram Ramanathan and Regina Rosales-Hain. Topology control of multihop wireless networks using transmit power adjustment. In *Proc. IEEE Infocom*, March 2000.

[69] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, Upper Saddle River, New Jersey, 1996.

[70] Samarth H. Shah and Klara Nahrstedt. Predictive location-based QoS routing in mobile ad hoc networks. In *Proceedings of IEEE International Conference on Communications*, 2002.

[71] Timothy Shepard. A channel access scheme for large dense packet radio networks. In *Proc. ACM SIGCOMM Conference (SIGCOMM '96)*, pages 219–230, August 1996.

[72] Prasum Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. CEDAR: A core-extraction distributed ad hoc routing algorithm. In *Proc. IEEE Infocom*, March 1999.

[73] Bernard Sklar. *Digital Communications: Fundamentals and Applications*. Prentice Hall, Upper Saddle River, New Jersey, second edition, 2000.

[74] James Almon Stevens. *Spatial Reuse Through Dynamic Power and Routing Control in Common-Channel Random-Access Packet Radio Networks*. PhD thesis, The University of Texas at Dallas, 1988.

[75] Audun Tornquist. Modular and adaptive ad hoc routing in Click. Master's thesis, University of Colorado, 2001.

[76] Rudi van Drunen, Jasper Koolhaas, Huub Schuurmans, and Marten Vijn. Building a wireless community network in the Netherlands. In *USENIX/Freenix Conference*, June 2003.

[77] Andreas Willig, Martin Kubisch, Christian Hoene, and Adam Wolisz. Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer. *IEEE Transactions on Industrial Electronics*, 49(6), December 2003.

[78] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. ACM Sensys*, November 2003.

[79] Mark Yarvis, W. Steven Conner, Lakshman Krishnamurthy, Jasmeet Chhabra, Brent Elliott, and Alan Mainwaring. Real-world experiences with an interactive ad hoc sensor network. In *Proceedings of the International Workshop on Ad Hoc Networking*, August 2002.

[80] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. ACM Sensys*, November 2003.

[81] Chenxi Zhu and M. Scott Corson. QoS routing for mobile ad hoc networks. In *Proc. IEEE Infocom*, June 2001.

Author Biography

Douglas S. J. De Couto was born in Bermuda in 1975. He attended the Saltus Grammar School in Bermuda, the Fessenden School in West Newton, Massachusetts, and the Hotchkiss School in Lakeville, Connecticut. He then studied at the Massachusetts Institute of Technology (MIT), where in 1998 he received his S.B. in Computer Science and Engineering, with a minor in Philosophy, and M.Eng. in Electrical Engineering and Computer Science. He received his Ph.D. from MIT in 2004, also in Electrical Engineering and Computer Science, with a minor in Finance. While at MIT he worked on a variety of computer systems projects, in particular implementing and studying ad hoc and multi-hop wireless networks. He also worked on dynamic voltage scheduling for low-power computing, and applications of the Global Positioning System for three-dimensional scene reconstruction. An avid competitive and recreational sailor, Mr. De Couto was captain of MIT's Varsity Sailing team, and was elected its MVP three years in a row. He also served two years as Commodore of the MIT Nautical Association.